N°d'ordre NNT : 2018LYSEI102

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
L'Institut National des Sciences Appliquées de Lyon

Ecole Doctorale N° 162
Mécanique, Énergétique, Génie civil, Acoustique

Spécialité/ discipline de doctorat :
Mécanique – Génie mécanique – Génie civil

Soutenue publiquement le 29/11/2018, par :
Matthieu Occelli

Explicit Dynamics IGA
LR B-Splines implementation in the Radioss solver

Devant le jury composé de :

BARANGER, Thouraya	Professeur, UCBL	Examinateur
BOUABDALLAH Salim	Docteur, Altair Engineering	Examinateur
BOUCLIER Robin	Maître de Conférences, INSA Toulouse	Invité
BREITKOPF Piotr	HDR, UTC Compiègne	Examinateur
ELGUEDJ Thomas	Professeur, INSA de Lyon	Directeur de thèse
EYHERAMENDY Dominique	Professeur, Ecole Centrale Marseille	Rapporteur
REALI Alessandro	Professeur, University of Pavia	Rapporteur

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 Fax : 04.72.43.16.87 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Viviane POLSINELLI Brigitte DUBOIS INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **viviane.polsinelli@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Une thèse est une période singulière de la vie synonyme d'intensité, de richesse en rencontres et en épreuves formatrices. Au moment de conclure ces travaux, de nombreuses personnes viennent à l'esprit et de chaleureux remerciements sont de rigueur.

Je voudrais tout d'abord remercier grandement mes encadrants de thèse, Thomas Elguedj et Salim Bouabdallah, pour leur aide, leur confiance, et leur bienveillance. Je suis ravi d'avoir travaillé en leur compagnie et malgré la distance, ils ont toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse. Leur expérience et leur rigueur scientifique ont grandement participé à la réussite de ces travaux. Leur encadrement a été enrichissant aussi bien du côté professionnel que du côté humain, et le fait de partager un nombre important de centres d'intérêt et de passions a largement contribué à alimenter nos échanges.

Je souhaite également remercier l'ensemble des membres de mon jury et leur exprimer ma reconnaissance pour avoir pris une partie de leur temps afin d'être présents. Tout d'abord à Thouraya Baranger de m'avoir fait l'honneur de présider ce jury. A Alessandro Reali et Dominique Eyheramendy qui ont eu la dure tâche d'être les rapporteurs de ma thèse. Leurs remarques constructives m'ont permis d'envisager mon travail sous d'autres angles et d'y apporter de nombreuses perspectives d'avenir. Pour tout cela je les remercie. A Piotr Breitkopf et Robin Bouclier pour leurs questions et conseils avisés. Je les remercie d'avoir accepté de participer au jury de soutenance. J'ai beaucoup apprécié les nombreux échanges qui ont suivit la soutenance et tâcherai de suivre leurs conseils dans ma carrière.

Je remercie toute l'équipe de doctorants du LaMCoS avec qui j'ai partagé ces années de thèse. Nos chemins se sont bien souvent croisés, mais nous avons noué des liens qui me sont chers aujourd'hui. Je tiens à remercier en particulier Tristan (le viandard), avec qui la science marche droit, Thibaut le cycliste le plus branché d'Austin, Tristan (le végan), Thomas, Alexis, Pierre, Nicolas, Zi, et tous les autres. Je leur souhaite de réussir leurs travaux de thèse et leur carrière professionnelle ou académique. Merci également à Isabelle pour son efficacité et sa disponibilité.

Je remercie Jean Michel Terrier, vice président d'Altair, pour avoir initié cette collaboration avec l'INSA de Lyon et le LaMCoS.

Je souhaite également remercier toute l'équipe de développement d'Altair, pour le support et les innombrables conseils qui m'ont permis d'avancer dans ces travaux. Toutes ces personnes ont contribué, par leur disponibilité et leur bonne humeur, à rendre cette thèse enrichissante et motivante au quotidien. Une pensée particulière pour Christophe, Benoît, Romain, David, Paméla, Maha, Maciek, qui m'ont beaucoup soutenu et surtout supporté dans les milles et un projets ou constructions que j'ai entrepris.

Je suis particulièrement fier d'avoir pu présenter mes travaux à ma famille, qui m'a fait l'immense joie de leur présence et de leur soutien le jour de la soutenance. Je n'oublie pas tout ceux qui n'ont pas pu être présents mais qui l'ont été durant ces années. Je tiens à dire un grand merci à mes parents, qui m'ont encouragé et porté depuis toujours et jusqu'à cette grande épreuve. C'est un accomplissement, pas seulement pour moi, dont chacun peut être fier.

Enfin, aucun mot ne peut exprimer ma reconnaissance à ma très chère Manon. Elle a su, tout au long de cette thèse, contenir mes ras le bol et m'encourager dans ma voie. Son soutien a été sans faille. Son sourire, sa tendresse et son amour me portent et me guident tous les jours.

Abstract

IsoGeometric Analysis has shown to be a very promising tool for an integrated design and analysis process. A challenging task is still to move IGA from a proof of concept to a convenient design tool for industry and this work contributes to this endeavor.

This work deals with the implementation of the IGA into Altair Radioss explicite finite element solver in order to address crash and stamping simulation applications. To this end, the necessary ingredients to a smooth integration of IGA in a traditional finite element code have been identified and adapted to the existing code architecture. A solid B-Spline element has been developed in Altair Radioss. The estimations of heuristic element and nodal stable time increment are explored to improve the accuracy of simulations and guarantee their stability. An existing contact interface has been extended in order to work seamlessly with both NURBS and Lagrange finite elements. As local refinement is needed for solution approximation, an analysis is made in terms of analysis suitability and implementation aspects for several Spline basis functions as Hierarchical B-Splines (HB-Splines), Truncated Hierarchical B-Splines (THB-Splines), T-Splines and Locally Refined B-Splines (LR B-Splines). The LR B-Spline basis is implemented. An improved refinement scheme is introduced and defines a set of analysis-suitable refinements to be used in Radioss. The refinement process of a regular coarse mesh is developed inside the solver. It allows the user to define a local refinement giving a set of instructions in the input file. The global solution is validated on industrial benchmarks, for validation cases conventionally used for industrial codes like stamping and drop test.

Keywords: Industrial solver, IsoGeometric Analysis, Volumetric approach, Local refinement, Explicit dynamic solution, Analysis-suitability

Résumé

L'analyse isométrique s'est révélée être un outil très prometteur pour la conception et l'analyse. Une tâche difficile consiste toujours à faire passer l'IGA de concept à un outil de conception pratique pour l'industrie et ce travail contribue à cet effort.

Ce travail porte sur l'implémentation de l'IGA dans le solveur explicite Altair Radioss afin de répondre aux applications de simulation de crash et d'emboutissage. Pour cela, les ingrédients nécessaires à une intégration native de l'IGA dans un code éléments finis traditionnel ont été identifiés et adaptés à l'architecture de code existante. Un élément solide B-Spline et NURBS a été développé dans Altair Radioss. Les estimations heuristiques des pas de temps élémentaires ou nodaux sont explorées pour améliorer l'efficacité des simulations et garantir leur stabilité. Une interface de contact existante a été étendue afin de fonctionner de manière transparente avec les éléments finis NURBS et de Lagrange. Un raffinement local est souvent nécessaire pour la bonne représentation de champs non linéaires tels que les champs de déformations plastiques. Une analyse est faite en termes de compatibilité pour l'analyse et de mise en œuvre pour plusieurs bases de fonctions Spline telles que les Hierarchical B-Splines, les Truncated Hierarchical B-Splines, les T-Splines et les Locally Refined B-Splines (LR B-Splines). Les LR B-Splines sont implémentés. Un schéma de raffinement est proposé et définit un sous-ensemble de raffinements adapté à leur utilisation au sein de Radioss. Le processus de raffinement d'un maillage initialement grossier et régulier est développé au sein du solveur. Il permet à l'utilisateur d'établir du raffinement local par un ensemble d'instructions à fournir dans le jeu de donnée de la simulation. La solution globale est validée sur des cas tests industriels, pour des cas de validation classiquement utilisés pour les codes industriels comme l'emboutissage et les tests de chute.

Mots clés: Solveur industriel, Analyse IsoGéométrique, Approche volumique, Raffinement local, Dynamique explicite, Compatibilité pour l'analyse

Résumé étendu

Introduction

L'Analyse IsoGéométrique (IGA) est une technique d'analyse numérique qui se révèle très prometteuse pour le design et l'analyse en remplaçant les éléments finis traditionnels par des éléments B-Splines et Non-Uniform Rational B-Spline (NURBS) [55, 28]. Les B-plines et NURBS ont initialement été choisies comme base de fonctions pour leur simplicité et leur utilisation dans l'industrie 3D, notamment pour la Conception Assistée par Ordinateur (CAO).

Le défi majeur aujourd'hui est d'amener l'IGA du stade de concept à un véritable outil pour l'industrie, et ce travail y amène une contribution. Les enjeux d'un code éléments finis industriel sont directement liés au monde de l'industrie. Les solutions apportées doivent être efficaces, générales, et doivent apporter des résultats de qualité tout en étant le plus robuste possible.

Altair Engineering est intéressé par l'intégration de l'IGA dans le solveur explicite Radioss. Les perspectives d'amélioration du pas de temps critique des simulations de phénomènes dynamiques tels que l'emboutissage ou le crash automobile a conduit à l'intégration native de l'IGA. La continuité des fonctions B-Spline et NURBS promet également d'obtenir une meilleur qualité de solution.

Les B-Splines et NURBS de degré bas, c'est à dire quadratique ou cubique, sont particulièrement intéressantes pour l'analyse. En effet, leur coût de calcul n'est pas beaucoup plus élevé comparé à des éléments finis conventionnels et elles permettent une amélioration importante de la qualité de solution.

Pour ces raisons, la thèse est axée sur l'implémentation d'un élément solide B-Spline et NURBS de degré bas, tel qu'il a été introduit par Cottrell et al. [28]. L'objectif est de traiter la robustesse et l'efficacité de ces bases de fonctions pour la simulation de problèmes dynamiques. C'est dans cette optique que les ingrédients nécessaires à l'intégration complète de l'IGA dans un code éléments finis traditionnel ont été identifiés et adaptés à la structure de code existante.

Un deuxième objectif est d'introduire une base de fonction Spline qui permet le raffinement local de modèles. Cette partie est liée à la compatibilité des technologies Spline existantes avec l'analyse et leur relative complexité d'implémentation.

Des B-Splines à l'analyse isogéométrique 3D de modèles localement raffinés

B-Splines, NURBS et analyse isogéométrique

Les B-Splines sont des fonctions polynomiales par morceaux, définies à partir d'un vecteur knot. Un vecteur knot Ξ est un ensemble de réels croissants définis dans l'espace paramétrique. Il est composé de $n+p+1$ valeurs, avec n le nombre de fonctions B-Spline d'ordre p:

$$\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}, \tag{1}$$

où $\xi_i \in \mathbb{R}$ est le i^{me} knot, i étant l'index du knot, avec $i = 1, 2, \ldots, n+p+1$.

Les B-Splines sont alors construites de manière récursive dans l'espace paramétrique à partir de combinaisons linéaires de fonctions B-Splines d'ordre inférieur, en commençant les fonctions constantes par morceaux ($p = 0$):

$$N_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi \leq \xi_{i+1}, \\ 0 \text{ sinon.} \end{cases} \tag{2}$$

pour $p \geq 1$, la base de fonction est définie par la relation récursive de Cox-de Boor [86] :

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{3}$$

.

Les fonctions NURBS sont construites à partir de fonctions B-Splines. Elles sont obtenues en attribuant un poids à chaque points de contrôle et selon la relation :

$$R_i^p(\xi) = \frac{N_{i,p}(\xi) w_i}{\sum_{i=1}^{n} N_{i,p}(\xi) w_i}, \tag{4}$$

où $N_{i,p}(\xi)$ est la i^{me} fonction B-Spline de degré p et w_i est le poids correspondant. Si tous les poids sont égaux, les NURBS deviennent des B-Splines.

Les fonctions B-Spline et NURBS permettent de construire des courbes, surfaces ou volumes. Soit un ensemble de n points de contrôle en 1D formant le polygone de contrôle, un ensemble de fonctions B-Spline ou NURBS $N_{i,p}, i = 1, 2, \ldots, n$ de degré p et un vecteur knot $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$. La courbe B-Spline est alors définie suivant la relation:

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) \boldsymbol{B_i}. \tag{5}$$

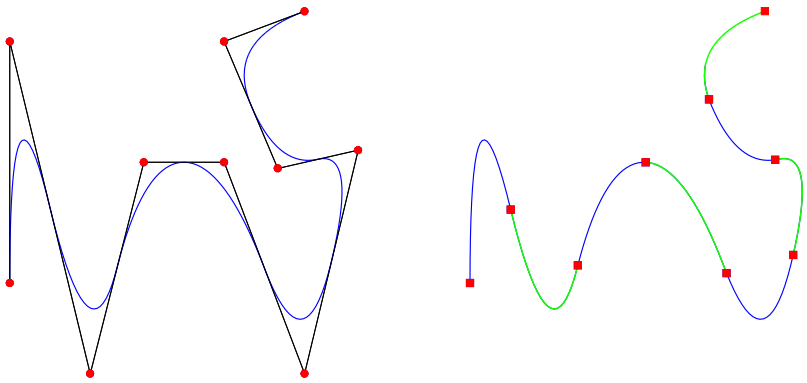


Fig. 1: Base de fonction B-Spline quadratique et polygone de contrôle correspondant au vecteur knot $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8\}$. a) Courbe B-Spline et points de contrôle modélisés par les rond rouges, b) positions des knot modélisés par les carrés rouges.

Le produit tensoriel global permet simplement de construire des surfaces et des volumes B-Spline ou NURBS.

Problématique du raffinement local

Plusieurs méthodes de raffinement associées aux B-Splines et aux NURBS ont été proposées par [28]. Le raffinement h- insère un nouveau knot dans le vecteur knot global, sans changer la géométrie de la courbe, de la surface ou du volume. Le raffinement p- augmente le degré polynomial de la base de fonctions, sans non plus changer l'objet final décrit. Enfin, le raffinement k-, augmente également l'ordre de la base de fonction sans augmenter la multiplicité des knots dans le vecteur knot. Cependant, le raffinement local des patchs B-Spline et NURBS est proscrit à cause du produit tensoriel intrinsèque à la formulation classique des B-Splines qui propage le raffinement jusqu'aux frontières du patch.

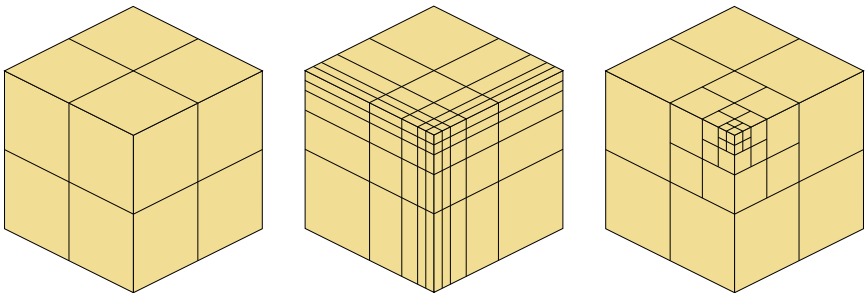


Fig. 2: Illustration de la problématique du raffinement local: (a) maillage initial, (b) Raffinement obtenu avec le produit tensoriel, (c) Raffinement local.

Plusieurs autres technologies Spline sont disponibles à ce jour dans la littérature et rendent le raffinement local possible.

Les B-Splines Hiérarchiques [101, 44, 91, 92], les B-Splines Hiérarchiques Tronquées [45, 66, 46], les T-Splines [95, 8] et les Locally Refined B-Splines [36, 57, 58] sont étudiées dans ce manuscrit. Ces formulations Spline permettent de conserver l'aspect local du raffinement à travers des notions de hiérarchies ou bien d'un produit tensoriel local.

L'analyse isogéométrique

L'analyse isogéométrique a été introduite par Hughes et al. [55]. Les intervalles knot dans le vecteur knot associé au patch définissent les éléments isogéométriques. Contrairement aux éléments finis, les fonctions B-Spline ont pour support plusieurs éléments isogéométriques et peuvent être continues sur un ensemble d'éléments. Un élément isogéométrique 3D contient dans son intervalle $(p+1)(q+1)(r+1)$ fonctions non nulles. Le patch est vu comme un macro-élément qui représente un domaine ou une partie de domaine physique.

Rapide revue de l'intégration de l'IGA

L'IGA a depuis été adaptée et utilisée dans de nombreux domaines tels que la mécanique des fluides, les vibrations, l'électromagnétisme, la biomécanique, les problèmes de coques, la représentation des contacts, les intéractions fluide/structure, et bien d'autres. Ce nouvelle base de fonction est bien entendu adaptée au formalisme des éléments finis, en procédant à certaines adaptations.

Durant les dernières années, des logiciels de recherche ont intégré l'IGA, dans Matlab avec GeoPDEs [29], dans OOFEM [90], en C++ avec Igatools [85], en C avec PetIGA [26], ou encore en Java [109].

Plusieurs implémentations de l'IGA dans des logiciels industriels ont été réalisées, par exemple dans LS-Dyna [14, 15] ou dans Abaqus [38, 68]. Dans LS-Dyna, la première implémentation a été réalisée pour une formulation coque. Plus tard, une implémentation pour une formulation solide a également été réalisée [17].

Implémentation des éléments isogéométriques dans le solveur explicite Radioss

Une formulation solide NURBS a été développée dans le code explicite Radioss en langage Fortran. Le fichier d'entrée contient l'ensemble des données nécessaires à la construction du maillage B-Spline ou NURBS. La connectivité des éléments est

écrite de façon similaire a des éléments classiques mais contient également les indices locaux reliant chaque élément isogéométrique aux vecteurs knot globaux. Les degrés polynomiaux ainsi que les vecteurs knot sont fournis dans un format spécifique. Les points de contrôle sont les homologues des nœuds et ont des poids associés.

Toutes les procédures de calcul des matrices masse, d'actualisation des masses volumiques, l'assemblage des forces interne ont été adaptées selon les spécificités des éléments isogéométriques.

Génération de modèle et post-traitement

La génération de maillages isogéométriques est toujours une tâche délicate et ne peut se faire qu'avec des outils adaptés. Des efforts ont été fournis dans cette direction pour développer des solutions robustes et exploitables [4, 6, 5, 69, 103, 107]. Un plugin développé dans le logiciel Rhinocéros permet de créer des géométries NURBS volumiques et des jeux de données compatibles pour une analyse avec le solveur Radioss.

Les résultats de simulations sont visualisés dans HyperView en utilisant un maillage éléments finis linéaires de projection. Chaque élément NURBS est représenté par un ensemble de 27 éléments finis hexaédriques, les grandeurs résultantes telles que les contraintes et les déformations étant interpolées aux nœuds du maillage de projection.

Application de chargements surfaciques

L'application des chargements surfaciques est assez simple à implémenter pour les élément isogéométriques. En effet, la paramétrisation des patchs B-Spline ou NURBS induit un maillage structuré pour lequel il est aisé de trouver les éléments d'une face externe. Pour chaque face externe de ces éléments, un vecteur normal est calculé d'après le produit vectoriel des composantes du Jacobien. La distribution des chargements surfaces sur les points de contrôle, connaissant l'expression de la surface externe et de sa normale, est similaire à ce qui est fait pour les éléments finis classiques. Un exemple d'application est donné représentant un cylindre épais soumis à une pression interne.

Methodes d'estimation du pas de temps critique

Tout comme les éléments finis classiques, l'implémentation de l'IGA dans un solveur explicite nécessite le calcul d'un pas de temps critique associé à ces éléments. Il est particulièrement important en termes de performance de réduire les temps de simulations, tout en assurant la stabilité du schéma d'intégration numérique. Le

comportement en hautes fréquences de l'IGA a été observé par le passé. Elle permet d'obtenir des pas de temps critiques plus grands. Les estimations du pas de temps critique des simulations provenant des éléments finis sont adaptées pour les éléments isogéométriques.

La longueur caractéristique

La méthode heuristique de la longueur caractéristique peut être utilisée pour l'IGA. Elle est égale au rapport du volume de l'élément par la plus grande de ses surfaces:

$$L_c = \frac{V_e}{max(S_e)}. \tag{6}$$

Les éléments isogéométriques peuvent cependant être très distordus et cette formulation peut conduire à de très mauvais pas de temps. Une autre méthode de calcul est donnée en utilisant pour le volume le volume minimum représenté par une bande de points d'intégration. L'aire associée est l'aire maximale des deux faces qui sont perpendiculaires à la bande de volumes.

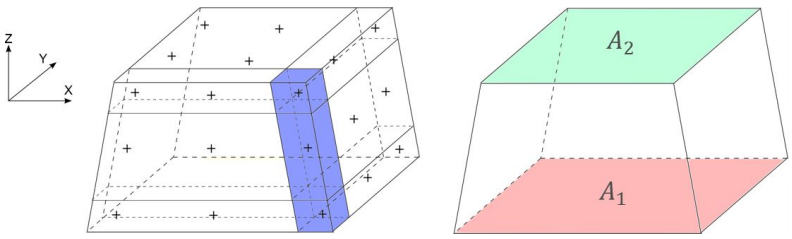


Fig. 3: Illustration du volume d'une bande de points d'intération et des surfaces associées pour un élément B-Spline ou NURBS quadratique.

La longueur caractéristique de l'élément est alors utilisée pour estimer le pas de temps critique, d'après la relation :

$$\Delta t \leq \frac{L_c}{c_0}. \tag{7}$$

avec c_0 la vitesse du son dans le matériau. Cette méthode est très conservative pour les élément intérieur avec une continuité maximale. Elle est cependant limitée par les éléments de bord qui ont une continuité plus basse.

La raideur nodale

Le pas de temps peut également être estimé en utilisant la méthode non heuristique de la raideur nodale. L'objectif est de calculer la valeur propre maximale ω_{max} du

système $\underline{\underline{M}}^{-1}\underline{\underline{K}}$ avec $\underline{\underline{M}}$ la matrice masse et $\underline{\underline{K}}$ la matrice de raideur. Le pas de temps estimé est alors égal à :

$$\Delta t = \frac{2}{\omega_{max}}. \tag{8}$$

La puissance itérée

La méthode de la puissance itérée constitue une troisième méthode d'estimation du pas de temps critique. Cette méthode approxime la plus grande valeur propre de manière itérative.

La puissance itérée n'est pas utilisée à chaque pas de temps, car elle est relativement couteuse. Elle est souvent couplée avec une deuxième méthode d'estimation, le pas de temps associé à la raideur nodale par exemple. Un ratio entre les deux pas de temps est calculé. Ce ratio varie faiblement comparé à la valeur du pas de temps pour la plupart des cas non linéaires. Lorsque la variation est trop importante, le pas de temps est corrigé par la puissance itérée. De cette façon, le coût de calcul associé à l'estimation du pas de temps est réduite.

Une simulation d'impact d'une barre de Taylor carrée est donnée pour comparer l'efficacité des estimations du pas de temps. Les temps et les coûts de calculs sont comparés pour un maillage isogéométrique et pour un maillage FEA. Les simulations montrent que le pas de temps pour les éléments isogéométriques est plus grand et permet de réduire fortement le temps de calcul. La méthode de la puissance itérée par rapport aux autres méthodes est de toute évidence la plus efficace pour atteindre la valeur maximale du pas de temps critique.

Contact isogéométrique

Depuis son introduction, l'IGA a montré la capacité de représenter de façon exacte des géométries complexes, mais aussi d'améliorer l'efficacité et la robustesse des simulations, comparé aux éléments finis classiques. L'une des applications qui peut bénéficier de l'IGA est la mécanique du contact. La continuité élevée des fonctions et l'exacte représentation de la géométrie constituent de potentiels avantages. Plusieurs formulations de contact ont été introduites dans le contexte de l'IGA. Leur complexité et leur simplicité d'implémentation dans les codes industriels sont à prendre en compte.

Dans le contexte de cette étude, une interface existante a été choisie pour y intégrer les modèles B-Spline et NURBS. Ce choix permet de bénéficier de l'efficacité d'algorithmes existant tels que le tri et la sélection des candidats au contact, en minimisant l'effort de programmation requis. L'interface "Type 7" de Radioss

a été choisie. Elle est basée sur une formulation nœud-segment selon l'approche maître/esclave avec pénalité non linéaire.

Représentation des surfaces B-Splines et NURBS

Dans le cas des B-Splines et NURBS, les points de contrôle ne sont généralement pas interpolants et ne peuvent pas être utilisés directement pour représenter les surfaces. La représentation des surfaces externes est faite au niveau des éléments par un ensemble de nœuds fictifs introduits et définissant des segments bilinéaires. Chaque face externe d'élément isogéométrique est recouverte par 16 points fictifs, c'est-à-dire 4 points par direction, selon un pas constant. Ces points qui sont physiquement collés au maillage isogéométriques sont mis à jour à chaque pas de temps, tout au long de la simulation.

Les 16 points fictifs de la face d'un élément sont directement utilisé comme nœuds esclaves. 9 segments sont construits pour chaque face d'éléments à partir des points fictifs pour le côté maître.

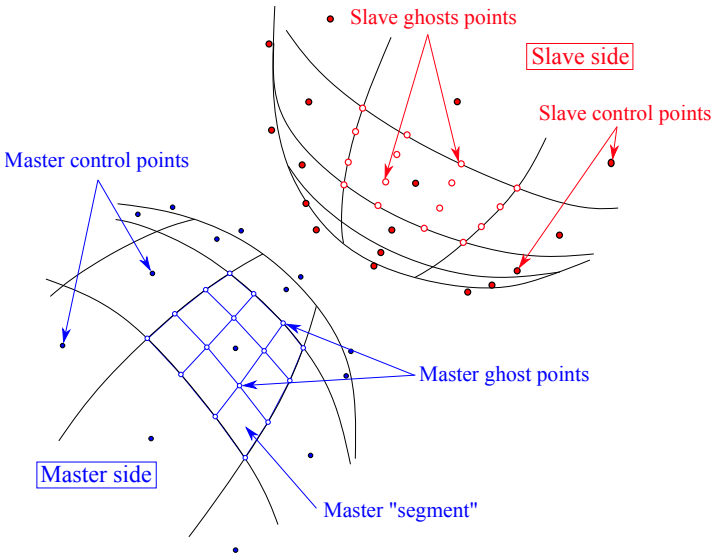


Fig. 4: Définition des surfaces maître et esclave pour deux géométries B-Spline quadratiques. La surface Spline et les points de contrôle sont tracés en noire. Le côté bleu représente le côté maître avec les segments construits à partir des points fictifs. Le côté rouge est le côté esclave avec les points fictifs associés.

L'utilisation des éléments isogéométriques dans l'interface de contact Type 7 utilise une structure de données propre. Les points fictifs sont stockés dans des tableaux spécifiques qui les rattachent à leur élément "parents". De cette façon, l'interface Type peut traiter indifféremment un ensemble de nœuds esclaves et de

segments maître. Les forces et raideurs de contact, lorsqu'elles sont renvoyées par l'interface, peuvent simplement être rapportées aux points de contrôle correspondant, grâce à ces structures de données.

Une simulation représentant le comportement dynamique d'un système came et soupape est proposée pour valider l'utilisation des éléments isogéométriques dans l'interface Type 7. Les résultats de cette simulation montrent la robustesse et l'efficacité de ces éléments. La facétisation des surfaces isogéométriques n'élimine pas les oscillations liées au contact discret, mais permet d'utiliser un maillage plus grossier tout en ayant une bonne représentation du comportement du modèle.

Raffinement local 3D : implémentation des LR B-Splines

Un raffinement local est nécessaire pour la bonne représentation de champs non linéaires comme les champs de déformations plastiques ainsi que pour la connexion de patchs. C'est l'ingrédient le plus récent ajouté à cette intégration.

Les différentes technologies Splines introduites plus haut peuvent être comparées en fonctions des propriétés nécessaires à leur intégration dans un code de calcul explicite, c'est à dire l'indépendance linéaire, la partition de l'unité, la non surcharge des éléments, l'existence d'une formulation 3D et les contraintes d'implémentation.

Parmi les différentes formulations disponibles, le choix s'est porté sur l'approche des Locally Refined B-Splines (LRBS). Cette base de fonctions utilise la méthode de raffinement h- de façon locale, permettant de limiter la propagation du raffinement des fonctions de forme à son minimum. Le choix a grandement été influencé par le fait qu'en 1D un élément n'a pas plus de $p+1$ fonctions non nulles avec p le degré polynomial des fonctions. Les structures d'un code élément fini sont habituellement dimensionnées et utilisées avec un nombre fixé de nœuds. Ce nombre de nœuds, ou dans le cas des LR B-Splines, dépend du type d'élément et du degré polynomial des fonctions.

La méthode des LR B-Splines, initialement introduite en 1D et 2D, est étendue en 3D. L'insertion de knot en 1D devient une insertion de mesh surface en 3D. Ces mesh surfaces sont définies par une valeur de knot particulière $\widehat{\xi}$ qui correspond au knot à insérer, et par une surface bornée donnée par $[\eta_1, \eta_2] \times [\zeta_1, \zeta_2]$, dans le plan formé par les deux autres directions paramétriques.

En analysant des raffinements LR B-Splines 2D, on peut se rendre compte qu'il existe un grand nombre de cas où les éléments peuvent être surchargés, c'est à dire qu'ils peuvent contenir plus de $(p+1)(q+1)(r+1)$ fonctions non nulles. Dans un

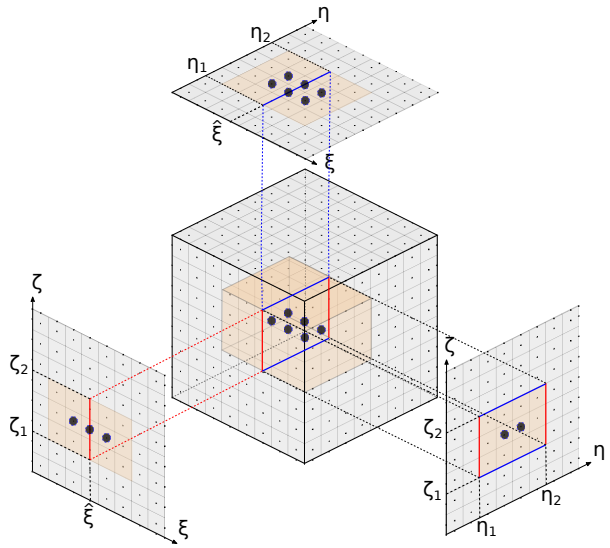


Fig. 5: Une mesh surface est définie par une valeur de knot spécifique $\hat{\xi}$ inséré et par une surface dans le plan des deux autres directions et bornée par $[\eta_1, \eta_2]$ et $[\zeta_1, \zeta_2]$. La surface de coupe est projetée sur les plans (ξ, η), (η, ζ) and (ζ, ξ) pour une meilleure visualisation.

raffinement hiérarchique du coin d'un patch 2D, l'élément du coin est coupé en deux dans chaque direction, puis l'élément du coin qui en résulte est également coupé, etc... Dans un tel modèle, un élément quadratique qui devrait contenir 9 fonctions non nulles peut en contenir 13. Des raffinements de cette sorte ne sont donc pas utilisables dans un code de calcul traditionnel.

Un schéma de raffiné qui a été nommé improved full span scheme, est proposé en 2D. Son extension en 3D définit un sous-ensemble de raffinements adaptés à leur utilisation au sein de Radioss. Les algorithmes de raffinement LR B-Splines existants sont également adaptés pour des modèles 3D et sont intégrés à un processus de raffinement d'un maillage initialement grossier et régulier au sein du solveur. Ce processus permet à l'utilisateur d'établir du raffinement local par un ensemble d'instructions à fournir dans le jeu de données de la simulation.

L'implémentation des LR B-Splines induit l'utilisation du produit tensoriel local plutôt que global pour l'estimation des fonctions de formes et de leurs dérivées. Cette estimation est donc plus couteuse pour les modèles raffinés et constitue à ce jour une limitation à son utilisation dans les codes industriels. Cette limitation n'est cependant pas uniquement liée aux LR B-Splines. Toutes les méthodes de raffinement local qui ont été étudiées sont aussi plus couteuses. Des efforts de recherche devront être faits pour accélérer ces calculs, en optimisant les algorithmes utilisés ou en stockant en mémoire un certain nombre de valeurs invariantes durant la sim-

ulation.

Exemples numériques - Applications

La solution globale est validée sur des cas tests industriels, pour des cas de validation classiquement utilisés pour les codes industriels comme l'emboutissage et les tests de chute. Ils illustrent les capacités de l'implémentation des B-Splines et des LR B-Splines. Deux de ces simulations sont brièvement présentées.

Tube cylindrique sous pression externe

La première illustre le flambement d'un cylindre en aluminium bouché à ses deux extrémités, placé dans une enceinte fermée et soumis à une pression externe. Cette simulation est basée sur les expérimentations de Farhat et al. [42]. Des simulations numériques ont également été réalisées en IGA avec des modèles coques par Benson et al. [14].

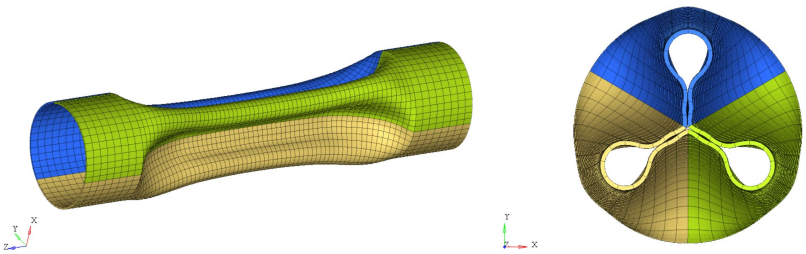


Fig. 6: Résultats de simulation pour le tube sous pression externe.

Le cylindre flambe selon le troisième mode. La déformée obtenue est similaire aux expérimentations et aux précédentes simulations. La valeur de pression critique est également proche des valeurs de référence.

Chute d'un carton

Cette simulation représente la chute d'un modèle de carton simplifié qui contient une charge à protéger. Différentes pièces ainsi que différents matériaux sont utilisés, tels que de la mousse, de l'aluminium, de l'acier. L'ensemble chute sur le coin.

Les résultats montrent que le carton et les mousses dans les coins se déforment fortement mais laissent le contenant intact. Le bilan des énergies mises en jeu permet également de s'assurer que l'énergie interne de ce contenant est bien inférieure à l'énergie interne de l'ensemble.

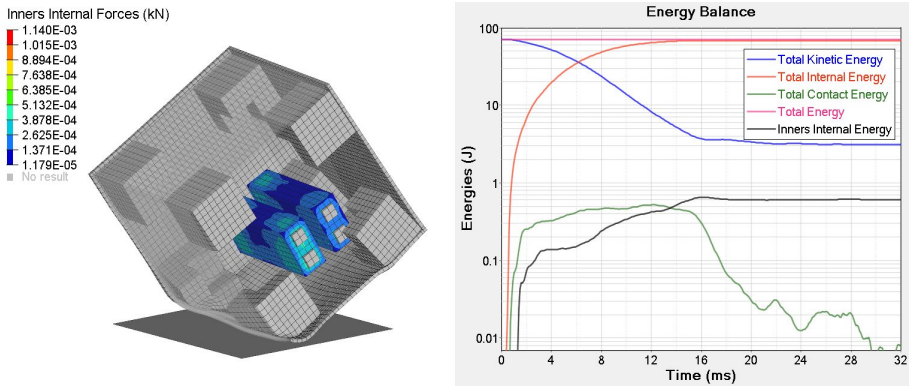


Fig. 7: Résultats de la simulation de la chute du carton et bilan d'énergies.

Conclusion

Cette thèse a été dédiée à l'implémentation de l'analyse isogéométrique dans un code explicite éléments finis industriel. Tous les ingrédients pour simuler des phénomènes dynamiques tels que l'emboutissage ou le crash ont été identifiés, modifiés et intégrés selon les spécificités de l'IGA. Plusieurs technologies Spline ont été mises en avant. Dans le cas particulier qu'est l'utilisation au sein du solveur Radioss, les LR B-Splines ont été choisies et implémentées. Un schéma de raffinement a été proposé et permet la définition de modèles localement raffinés, compatibles pour l'analyse.

L'utilisation de l'IGA dans Radioss a amélioré la qualité des résultats obtenus comparé aux éléments finis conventionnels. Nous avons été en mesure de formuler des estimations pour le nouveau type d'élément qu'est l'élément isogéométrique. Le pas de temps des simulations obtenu avec les éléments isogéométriques est plus grand. Il constitue un argument très fort à mettre en avant pour promouvoir les formulations Spline dans les solveurs, notamment explicites.

La technologie des LR B-Splines est actuellement plus couteuse et devra être poursuivie par un travail lié à la performance des algorithmes employés. Plusieurs pistes d'améliorations sont déjà à l'étude, comme la possibilité de conserver en mémoire des données invariantes qui sont à ce jour recalculées de nombreuses fois durant les simulations.

Les optiques de recherche et d'amélioration de la solution sont nombreuses. La surface réelle Spline pourrait être utilisée dans les formulations de contact. La qualité de représentation du contact serait grandement améliorée. Une formulation coque pourrait également être développée dans Radioss, pour adresser des problèmes de structures minces. A plus long terme, des pistes de ruptures peuvent être étudiées : les méthodes de collocation EF-IGA, l'utilisation de vecteurs knot ouverts ou l'intégration sélective.

Contents

List of Figures

List of Tables

Introduction

Since its introduction, IsoGeometric Analysis has demonstrated a high potential gain with respect to efficiency, quality and accuracy in analysis by replacing traditional finite elements by B-Spline and Non-Uniform Rational B-Spline (NURBS) elements. NURBS functions were initially chosen as a basis due to their relative simplicity and their use in the graphics industry: Computer Aided Design (CAD) and Animation. They provide more accurate modeling of complex geometries and exactly represent shapes such as circle, cylinders, with coarse level of discretization. They thus allow to eliminate geometrical errors frequently found for classical finite element models.

Isogeometric analysis has since been used and adapted in many fields such as fluid mechanics, vibrations, electromagnetism, biomechanics, shell problems, contact representations, fluid/structure interaction problems and many more. These new basis functions are of course usable in a finite element formalism with some modifications.

Several IGA implementations in industrial software were thus also done, e.g. in LS-Dyna and Abaqus. IGA implementations were done at the beginning for shells, providing good results and accuracy, benefiting in the same time from less complexity compared to a solid model. Then, attempts to implement solid models have more recently been done.

These studies and implementation attempts showed B-Splines and NURBS capabilities and accuracy, in particular for explicit transient analysis. In general in FEA, increasing the order of the basis functions increases accuracy but it also results in the appearance of strange frequencies at the very end of the spectrum. Their number and magnitude increase with the degree. These frequencies can be ignored when using an implicit time integration algorithm but in explicit transient analysis it would be detrimental for the accuracy and the quality of results. Indeed, the frequencies of the highest modes are grossly overestimated and stability would necessitate an unacceptably small time increment.

NURBS are more accurate throughout the entire spectrum. They do not diverge in transcient analysis and allow to keep a high time increment.

However, from the analysis point of view, the tensor product structure of NURBS can be seen as a drawback as it prevents local refinement that is commonly used with classical FEA. Many different technologies have addressed the problem of local refinement, in particular Hierarchical B-Splines, Truncated Hierarchical B-Splines, T-Splines and Locally Refined B-splines. They were introduced and subsequently improved in the past decade focusing on stability, linear independence and partition of unity properties. Research work is still ongoing for most of these Spline basis to bring a technology with the desired analysis properties and the ability to locally refine meshes.

The aim of this thesis

From this point of view, Altair Engineering is interested in integrating IGA into the Radioss explicit solver. The prospects for improving the time increment of simulations of dynamic phenomena such as stamping or automotive crash led to the native integration of IGA. The continuity of the B-Spline and NURBS functions also promise to achieve a higher quality of results than with conventional finite elements.

Low degree B-Splines and NURBS, i.e., quadratic or cubic, are particularly attractive for analysis. Indeed, their computationnal cost is not so much higher than compared to conventional finite elements and they allow a great improvement of the quality of solution.

For these reasons, the thesis is focused on implementing a low degree B-Spline and NURBS solid element in Radioss, as it was introduced by Cottrell et al. [28]. The aim is to deal with the robustness, the efficiency and the quality of solution provided by Spline basis function in order to address crash and stamping simulation applications.

A second objective is to introduce a Spline basis that allows locally refined models. This part is more related to the analysis-suitability of existing Spline basis and their easy to implement aspect.

Outline of this thesis

The thesis is divided into 4 individual chapters. The first one is a state of the art and will re-introduce B-Spline and NURBS concepts with their application to analysis including refinement strategies. Several Spline basis allowing local

refinement will also be detailed and compared according to some targeted properties and implementation constraints. Others ingredients to an integration in an explicit solver like contact formulations will be reviewed, describing different existing approaches. Finally, a brief review of IGA implementation attempts will be given, showing the scientific and industrial fields of interest for this recent method.

The second chapter gives all that is needed to succeed in a native implementation of NURBS based IGA in the context of an explicit solver. The mesh generation and the post processing and visualization are adressed in the case of our usage in Radioss. Some empirical methods to estimate the time increment are detailled. Some of them are directly generalized from FEA whereas others have to be revised in order to be adapted to IGA. The modification of a contact interface which can work with both FEA and IGA is also described. The efficiency and the quality of the obtained results are compared on a classical benchmark.

The third chapter focuses on the Locally Refined B-Splines implementation and usage in the trivariate case. A new general refinement scheme for 2D and 3D LR B-Splines with arbitrary shape of the refined area is introduced. The required properties to obtain a suitable basis are ensured, especially the non-overloading property. The obtained solution really improves the first implementation approach and gives the opportunity to address many others IGA aspects. Its use in Radioss is described and adressed efficiency and computationnal cost.

The fourth chapter is dedicated to more complex numerical simulations and will further illustrate the method. The performance of the LR B-Splines based IGA will be adressed in some industrial simulation topics like stamping, droptest or crash simulations.

Finally general conclusions will be drawn as well as short and long term prospects.

From B-Spline to 3D locally refined IGA

Contents

1.1 B-Splines, NURBS and IGA basics

1.1.1 Knot vector and B-Spline basis functions

A knot vector Ξ is a set of non-decreasing real numbers, defined in the parametric space. It is composed of $n+p+1$ values, with n the number of univariate B-Spline basis functions of order p.

$$\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}, \tag{1.1}$$

where $\xi_i \in \mathbb{R}$ is i^{th} knot, i is the knot index and $i = 1, 2, \ldots, n+p+1$. Knots divide the parametric space in knot spans $[\xi_i, \xi_{i+1}]$ and the global interval $[\xi_1, \xi_{n+p+1}]$ defines a patch. A knot vector is called uniform if its knots are uniformly spaced, and non-uniform otherwise. It is also called non-periodic or open if its first and last knots are repeated $p+1$ times, periodic or closed otherwise. Different types of knot vectors are given in Table 1.1 for quadratic Splines. In the following, for isogeometric analysis, only open knot vectors will be used. It allows notably to assemble patches like are assembled classical finite elements and ease the establishment of boundary conditions.

	Open/non-periodic	Closed/periodic
Uniform	$\{0,0,0,1,2,3,4,4,4\}$	$\{-1,-1,0,1,2,3,4,5,6\}$
Non-uniform	$\{0,0,0,1,3,4,4,4\}$	$\{-2,0,1,2,4,6\}$

Table 1.1: Different types of knot vectors for quadratic Splines.

1.1.1.1 B-Spline basis functions

B-Splines are piecewise polynomial functions, defined from a knot vector Ξ. They are constructed recursively in the parametric space from linear combinations of lower order B-Spline functions, beginning with piecewise constants ($p = 0$) :

$$N_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi \leq \xi_{i+1}, \\ 0 \text{ otherwise.} \end{cases} \tag{1.2}$$

For $p \geq 1$, the basis is defined by the Cox-de Boor recursion formula [86]:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{1.3}$$

The constructed basis functions for a uniform knot vector $\{0, 1, 2, 3, 4, ...\}$ is given in Figure 1.1. We can notice that for $p = 0$ and $p = 1$, the basis functions are identical to the classical Lagrange polynomial basis functions.

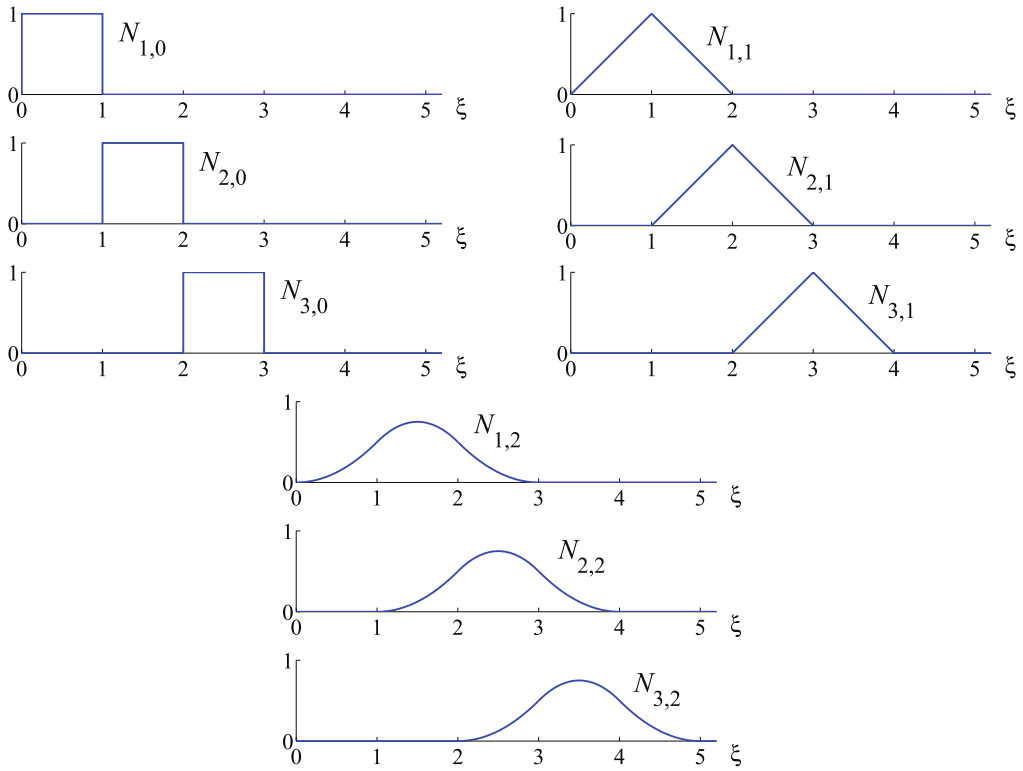
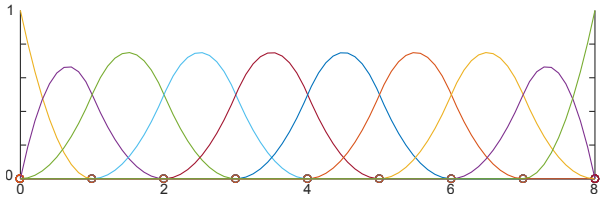


Fig. 1.1: Basis functions with $p = 1, 2$ or 3 for a uniform knot vector $\{0, 1, 2, 3, 4, ...\}$, from [28].

Local point of view B-Spline basis functions defined by the global knot vector can be seen with a local point of view. A knot vector of size $n + p + 1$ will generate n linearly independent basis functions of degree p on $p + 2$ knots for support. Each basis function support is a part of the knot vector, where the corresponding function is non-zero. Reciprocally, a knot span $[\xi_i, \xi_{i+1}]$ is covered by $p + 1$ B-Spline basis functions. According to these considerations, a given open uniform knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8\}$ will define 10 quadratic B-Spline functions with 10 so called local knots and 8 knot intervals, see Figure 1.2:

Fundamental properties Below are listed some of the fundamental properties of B-Spline basis function:

- Continuity: across knots, basis functions will be C^{p-m} where p is the polynomial degree and m is the multiplicity of the knot. B-Spline basis functions constructed from an open uniform knot vector are interpolatory, i.e., they are C^0 at the extremities of the knot interval $[\xi_1, \xi_{n+p+1}]$, but are not in general at the interieur knots. In the case of the basis constructed from the knot vec-



$$
\begin{aligned}
\Xi &= \{0,0,0,1,2,3,4,5,6,7,8,8,8\}\\
\Xi_1 &= \{0,0,0,1\},\\
\Xi_2 &= \{0,0,1,2\},\\
\Xi_3 &= \{0,1,2,3\},\\
\Xi_4 &= \{1,2,3,4\},\\
\Xi_5 &= \{2,3,4,5\},\\
\Xi_6 &= \{3,4,5,6\},\\
\Xi_7 &= \{4,5,6,7\},\\
\Xi_8 &= \{5,6,7,8\},\\
\Xi_9 &= \{6,7,8,8\},\\
\Xi_{10} &= \{7,8,8,8\}.
\end{aligned}
$$

Fig. 1.2: All quadratic basis functions generated by the knot vector $\Xi = \{0,0,0,1,2,3,4,5,6,7,8,8,8\}$. Each individual basis function can be described using a local knot vector of $p+2$ knots each.

tor $\Xi = \{0,0,0,1,2,3,4,5,6,7,8,8,8\}$, none of the middle knots is repeated. Their multiplicity is then minimum ($m = 1$).

- A B-Spline basis constructed from an open knot vector has the partition of unity property:

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \qquad \forall \xi \in [\xi_0, \xi_{n-p}] \tag{1.4}$$

- A B-Spline basis is linearly independent:

$$\sum_{i=1}^{n} c_i N_{i,p}(\xi) = 0 \Leftrightarrow c_i = 0, i = 1, ..., n, \qquad \forall \xi \in [0,1] \tag{1.5}$$

- A B-Spline basis has a compact support, i.e., the support of $N_{i,p}$ is included in $[\xi_i, \xi_{i+p+1}]$.

1.1.2 B-Spline curves, surfaces and volumes

B-Spline curves can be built in $\mathbb{R}^d$ by linear combination of B-Spline basis functions. Coefficients $B_i \in \mathbb{R}^d, i = 1,2,\ldots,n$ are called control points. They are analogous to nodes for finite element analysis and define the control polygon. Given n functions $N_{i,p}, i = 1,2,\ldots,n$ of degree p and a knot vector $\Xi = \{\xi_1,\ldots,\xi_{n+p+1}\}$, a B-Spline curve is defined by:

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\boldsymbol{B}_i. \tag{1.6}$$

An example of B-Spline curve and the defined mesh for the previous set of functions is given in Figure 1.3.

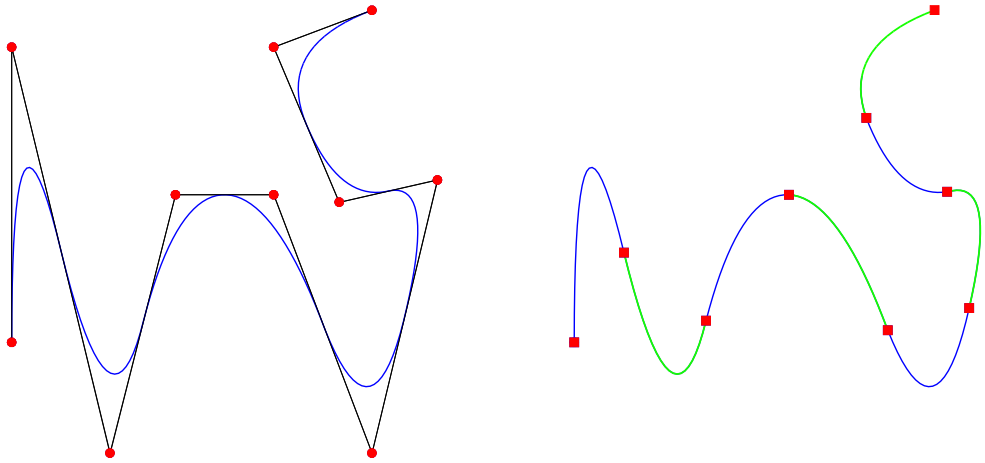


Fig. 1.3: Quadratic B-Spline curve and control polygon corresponding to the knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8\}$. a) B-Spline curve and control points modeled by the red circles, b) knot positions modeled by the red squares.

1.1.2.1 B-Spline surfaces and volumes

Given a control polygon $B_{i,j} \in \mathbb{R}^d, i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m$ and knot vectors for each parametric direction $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$ and $\Psi = \{\eta_1, \ldots, \eta_{\mathrm{m+q+1}}\}$ respectively of degree p and q, a B-Spline surface can be constructed using the global tensor product generalization of univariate B-Spline functions as follows:

$$\boldsymbol{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta) \boldsymbol{B}_{i,j}. \tag{1.7}$$

Given a control polygon $B_{i,j,k} \in \mathbb{R}^d, i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m,\ k = 1, 2, \ldots, l$ and knots vectors $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$, $\Psi = \{\eta_1, \ldots, \eta_{\mathrm{m+q+1}}\}$ and $Z = \{\zeta_1, \ldots, \zeta_{n+p+1}\}$, respectively of degree p, q and r, a B-Spline volume can be constructed in a similar way:

$$\boldsymbol{V}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) \boldsymbol{B}_{i,j,k}. \tag{1.8}$$

2D and 3D basis functions have the same fundamental properties that 1D basis functions.

1.1.3 Non-Uniform Rational B-Splines

NURBS are constructed from B-Splines. They can be obtained by weighting the B-Spline functions, assigning a weight to each of the control points. More

specifically, NURBS functions in $\mathbb{R}^d$ are built from projective transformations of B-Splines in $\mathbb{R}^{d+1}$, defined as:

$$R_i^p(\xi) = \frac{N_{i,p}(\xi) w_i}{\sum_{i=1}^{n} N_{i,p}(\xi) w_i}, \tag{1.9}$$

where $N_{i,p}(\xi)$ is the i^{th} B-Spline basis functions of degree p and w_i is the corresponding weight. If all the weights are equal, NURBS become B-Splines.

Analogously, NURBS curves are built from a linear combination of NURBS functions and the global tensor product is used to obtain NURBS surfaces and volumes:

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} R_{i,p}(\xi) \boldsymbol{B}_i, \tag{1.10}$$

$$\boldsymbol{V}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{o} N_{i,p}(\xi) M_{j,q}(\eta) R_{k,r}(\zeta) \boldsymbol{B}_{ijk}. \tag{1.11}$$

The main advantage of NURBS compared to B-Splines is their ability to represent more geometric entities. NURBS can precisely represent conic sections, such as circles and ellipses. They inherit fundamentals properties from B-Splines, i.e.,:

- The NURBS basis functions have the same continuity and the same support as their B-Spline counterparts.
- They form a partition of unity and are linearly independent.

1.1.4 B-Spline basis function refinement

B-Spline basis function can be enriched in several ways leaving the geometry and its parameterization intact. B-Spline refinement methods differ from finite element one, allowing to increase the order of the basis or inserting new knot values but also increasing or decreasing the continuity of the basis. In the following are presented these refinement strategies.

1.1.4.1 h- and p- refinement

h- refinement: knot insertion h-refinement is based on the knot insertion procedure. Knots can be inserted in the existing knot vector without changing a curve geometrically or parametrically. Given a knot vector $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$, and $\overline{\xi} \in [\xi_k, \xi_{k+1}]$ a new knot to be inserted, the new basis of $n+1$ functions is created recursively using Equation (1.12) and Equation (1.13), with the new knot vector $\Xi = \{\xi_1, \ldots, \xi_k, \overline{\xi_k}, \xi_{k+1}, \ldots, \xi_{n+p+1}\}$. The $n+1$ modified control points $\{\overline{B_1}, \ldots, \overline{B_{n+1}}\}$ are defined from original control points $\{B_1, \ldots, B_n\}$ with the

following, see Figure 1.4:

$$\overline{\boldsymbol{B}_i} = \alpha_i \boldsymbol{B}_i + (1 - \alpha_i)\boldsymbol{B}_{i-1}, \tag{1.12}$$

with

$$\alpha_i = \begin{cases} 1 \; si \; 1 \leq i \leq k-p, \\ \frac{\bar{\xi} - \xi_i}{\xi_{i+p} - \xi_i} \; si \; k-p+1 \leq i \leq k, \\ 0 \; si \; 1 \leq i \leq n+p+2. \end{cases} \tag{1.13}$$

An example of h- refinement is presented in Figure 1.4. The original curve is the same as in Figure 1.3 composed by quadratic B-Spline functions with the knot vector of Figure 1.2. The original curve, the basis functions and the control mesh are shown on the left. Each intermediate knot is inserted. The new knot vector is $\Xi = \{0, 0, 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8, 8\}$. Refined curve and basis functions are shown on the right. The curve is unchanged, there are more control points and more knot intervals.

Insertion of knot values is partly similar to the classical FEA refinement. The knots already present in the knot vector can be repeated. However, this reduces the continuity of the basis functions at the knot. The repeating of existing knot values does not have any equivalent in FEA.

p- refinement: order elevation p- refinement is done by order elevation. The polynomial degree of the basis functions can be increased without changing the curve, neither parametrically nor geometrically. Each value Ξ in the knot vector must be repeated, in order to preserve the discontinuity of the derivatives of the curve whose order is raised. The number of new control points depends on the multiplicity of existing knots. As for the knot insertion, the solution space defined by the new basis contains the space defined by the original basis.

For the p-refinement case, the order of the curve is increased. The multiplicity of knots is increased by one. The position and the number of control points changes, but the curve remains the same. Although there is the same number of functions, the positions of the new control points obtained from refinements h and p are different.

1.1.4.2 k- refinement: order and continuity elevation

A higher order alternative strategy is possible. It takes advantage of the fact that knot insertion and order elevation do not commute. If a single knot ξ is inserted be-

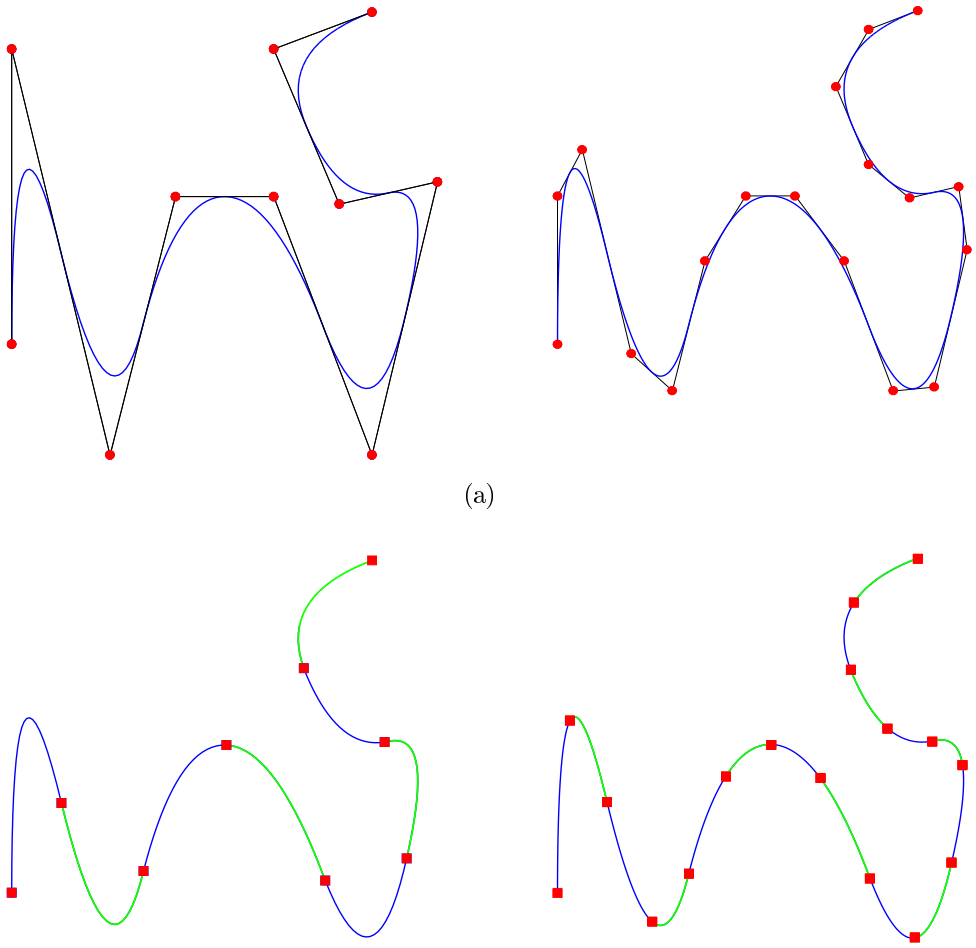


Fig. 1.4: h-refinement of a quadratic B-Spline basis. The coarse basis is plotted on the left, the refined one on the right. a) B-Spline curve and control points modeled by the red circles, b) knot positions modeled by the red squares.

tween two distinct knots in a curve of polynomial degree p, the number of continuous derivatives of the basis functions in ξ is $p-1$. Subsequently, if the order is raised to q, the multiplicity of each knot is increased to maintain a C^{p-1} continuity of knot functions ξ. If, instead, the order is raised to q and then inserting a single knot ξ, the basis function has $q-1$ continuous derivatives at ξ. This strategy is called k-refinement. k-refinement produces fewer functions with a higher continuity. This results in a smaller number of control variables and therefore degrees of freedom, for the mesh and an identical order of approximation. It has no counterpart in finite elements.

1.1.5 NURBS based and B-Spline based IGA

IsoGeometric Analysis was introduced by Hughes et al. [55]. Few elements of this paper are presented here.

1.1.5.1 Patches and elements

In the previous sections we introduced NURBS and B-Spline basis functions and their properties within a patch, in order to use them in IGA. Introducing IGA in a finite element context required to define what will be called an isogeometric element. Several points of view exist in the literature: it was suggested to liken the element to a patch in [62], or rather to a knot interval within the patch in [28]. This representation of isogeometric element is more in accordance with a classical finite element one in terms of inherent data. It will be easier to introduce that definition of isogeometric element in the existing data structures usually used for finite elements, particularly for commercial FE software.

A patch will be seen as a macro-element, regrouping several elements and representing an entire physical domain or a part of it. Many simple domains can be represented by only one patch. Instead of using subdomains of the physical domain, patches play the role of subdomains in IGA. The parametric space is local to patches and within each patch material models are assumed to be uniform. Knot vectors in the parametric space are then used to define the elements, see Figure 1.5:

Isogeometric element definition. In this work, knot spans defined by $[\xi_i, \xi_{i+1}]$ in 1D, $[\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$ and $[\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}] \times [\zeta_k, \zeta_{k+1}]$ respectively for 2D and 3D will define respectively 1D, 2D or 3D elements. These knot spans are overlaped by $p+1$ non-zero B-Spline basis functions, respectively by $(p+1)(q+1)$ and $(p+1)(q+1)(r+1)$ non-zero basis functions in 2D and 3D. The so-called variable *nctrl* is introduced and is equal to this number of non-zero basis functions.

We can then consider local mappings and rewrite Equation (1.14) in 1D and Equation (1.15) in 3D:

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{nctrl} N_i(\xi)\boldsymbol{B}_i, \tag{1.14}$$

$$\boldsymbol{V}(\xi,\eta,\zeta) = \sum_{i=1}^{nctrl} N_i(\xi,\eta,\zeta)\boldsymbol{B}_i, \tag{1.15}$$

where $N_i(\xi)$ or $N_i(\xi,\eta,\zeta)$ are the B-Spline or NURBS basis functions.

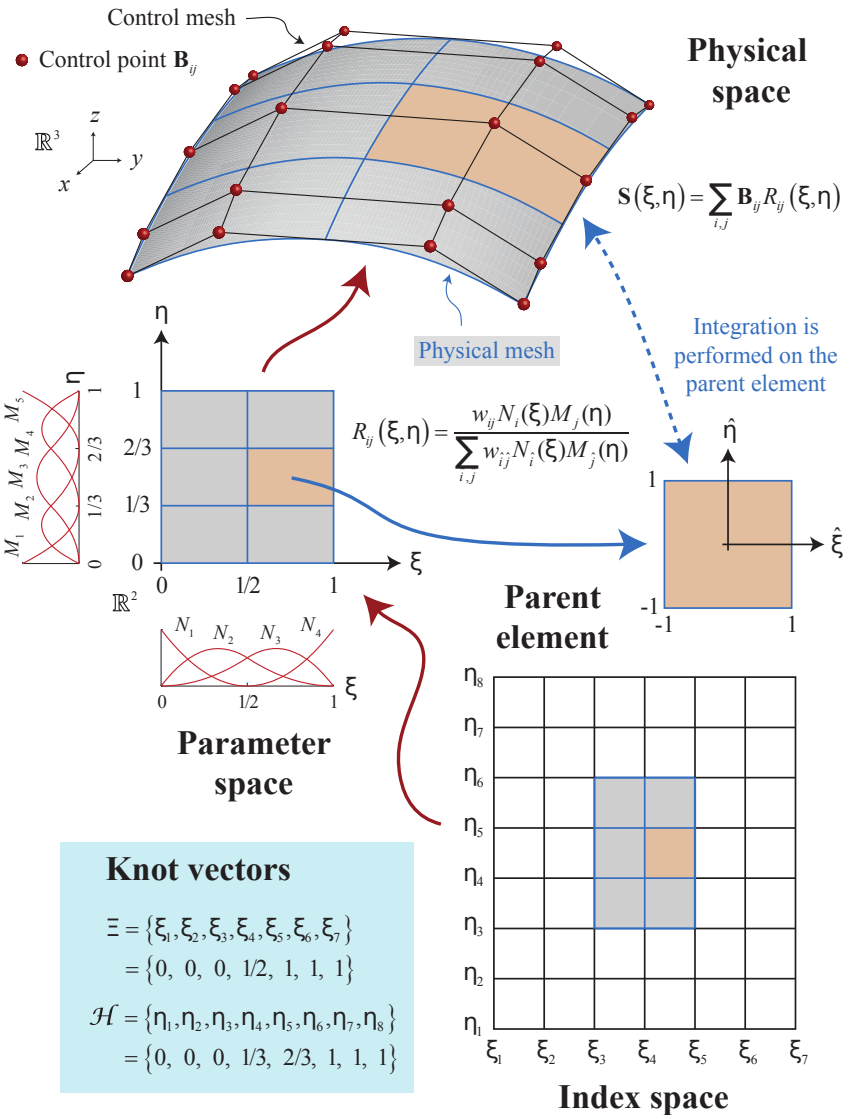


Fig. 1.5: Parametric space. We can see how the physical domain is represented by one patch consisting of the control net and the mapping from the parametric space on to the physical space. Illustration taken from [28].

1.1.5.2 Summary of IGA characteristics

For B-Spline or NURBS based IGA, there are two notions of meshes, the control mesh and the physical mesh, contrary to FEA where there is only one. The control points define the control mesh, and the control mesh interpolates the control points. The control mesh consists of multilinear elements, in two dimensions they are bilinear quadrilateral elements, and in three dimensions they are trilinear hexahedra. The control mesh does not conform to the actual geometry but controls

it like a scaffold. The control mesh has the look of a typical finite element mesh of multilinear elements. The control variables are the degrees of freedom and they are located at the control points. They may be thought of as generalized coordinates. Control elements may be degenerated to more primitive shapes, such as triangle and tetrahedra. The control mesh may also be severely distorted and even inverted to an extent, while at the same time, for sufficiently smooth NURBS, the physical geometry may still remain valid, which is not the case with finite elements. B-Splines and NURBS have properties as they serve as a basis for IGA. These properties are as follows:

- The mesh of a B-Spline or NURBS patch is defined as the tensor product of knot vectors. For example, in three dimensions, a mesh is given by $\Xi \times \Psi \times Z$.
- Knot intervals span the domain in what is called isogeometric elements.
- The support for each basis function consists of a small number of elements.
- The control points associated with the basis functions define the geometry.
- The isoparametric concept is invoked. The associated coefficients with the basis functions are the degrees of freedom or control variables.
- Three refinement strategies are available: the equivalents to h- and p- refinements, as well as a new refinement of higher order, k- refinement.
- The matrices and vectors associated with the elements built from isoparametric B-Splines or NURBS are assembled in matrices and global vectors in the same way as in finite elements [54].
- Dirichlet boundary conditions are applied to control points. The homogeneous conditions are satisfied pointwise. In the case of inhomogeneous conditions, the boundary conditions shall be approximated by functions included in the B-Spline or NURBS function space. The condition is then satisfied in a strong but approximate way. One can also consider imposing these conditions in a weak way [7]. Neumann conditions are naturally satisfied and very similar to standard finite elements [54].

1.1.6 B-Splines and NURBS: the local refinement issue

Given the fact that the patch-wise global tensor product is an intrinsic property of B-Splines and NURBS in 2D and 3D, the insertion of new knots in a knot vector (h- of k- refinement) is propagated all along the others parametric directions.

An example of a 3D refinement is given in Figure 1.6, where the corner is recursively refined. The refinement looses its locality with the classical knot insertion. This limitation for 2D and 3D refinements is well-known for B-Splines and NURBS.

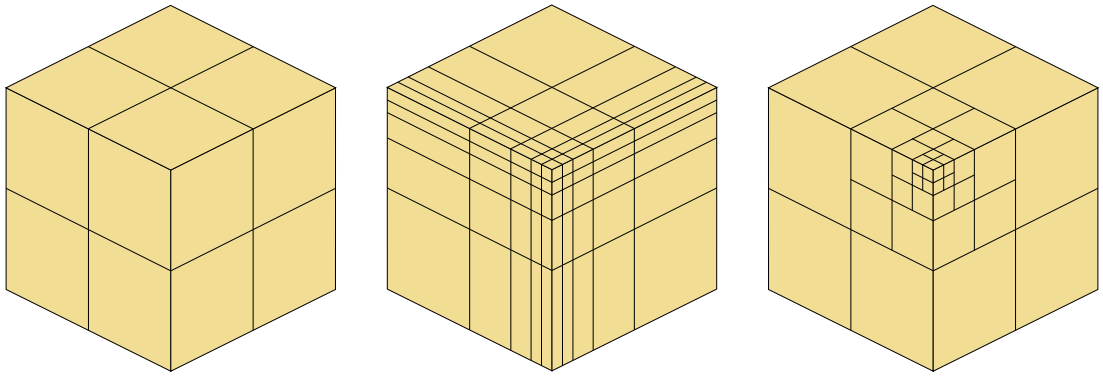


Fig. 1.6: Illustration of the local refinement issue for B-Splines: (a) initial mesh, (b) tensor product refinement, (c) targeted local refinement.

Other Spline functions are then needed to remove the limitation of classical B-Splines and to achieve true local refinement. In the following sections we will introduce several Spline technologies that allow for local refinement. For simplicity and illustration purpose, we will use the same 1D example introduced here after.

Definition of the area to be refined: Going back to B-Spline basis functions, one can define an initial coarse B-Spline basis with the same global knot vector $\Xi = [0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8]$. We assume that one function of the initial level is selected for refinement. The area to be refined is set between knots $\xi = 4$ and $\xi = 7$, i.e. for 3 elements, see Figure 1.7.

1.2 Hierarchical B-Splines

Hierarchical B-Splines were introduced and used for IGA by Vuong et al. [101], Giannelli and Jüttler [44] and by Schillinger et al. [91, 92]. A finer basis is generated subdividing the coarse knot vector by two, adding each intermediate knot value, see Figure 1.8.

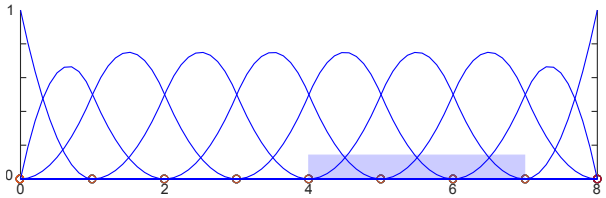


Fig. 1.7: Quadratic B-Spline basis function, defined from $\Xi = [0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8]$ as the coarse level of function. The area to be refined is colored in light blue, between knots $\xi = 4$ and $\xi = 7$.

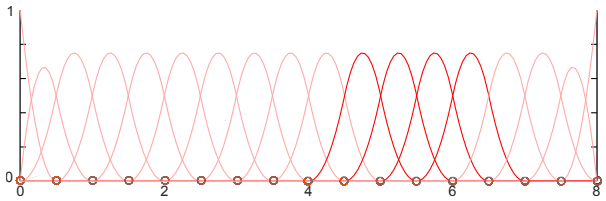


Fig. 1.8: Quadratic B-Spline basis function defined with a knot vector composed by the same knot than Ξ adding each intermediate knot value. Finer functions non zero on the area to be refined are colored in dark red.

The hierarchical B-Spline refinement is quite simple. The coarse basis function fully contained in the refinement area is replaced by the $p + 2$ non-zero finer functions on this area, see Figure 1.9. This gives a hierarchy of nested domains. Somes functions from the coarser domains are removed, and functions of the finer are added. The refinement procedure starts from the coarser level and generates one or several nested finer levels, depending on the depth of refinement needed. The definition, see Equation (1.16), ensures that the correct functions are always selected to include in the basis.

- Initialization : $H^0 = \{N \in \nu^0 : supp\, N \neq \emptyset\}$
- Recursive case : $H^{l+1} = H_A^{l+1} \cup H_B^{l+1}$ for $l = 0, \ldots, M-1$, where
  $H_A^{l+1} = \{N : N \in \nu^l : supp\, N \not\subseteq \Omega^{l+1}\}$
  $H_B^{l+1} = \{N : N \in \eta^{l+1} : supp\, N \subseteq \Omega^{l+1}\}$ (1.16)
- $H = H^M$

As a result of the definition, the Hierarchical B-Spline basis and the associated spaces have the following properties, see Vuong et al. [101], Giannelli and Jüttler [44] for details:

- The functions in H are linearly independent.
- The spaces spanned by the basis are nested. One can notice than for this refinement method, partition of unity is lost. Normalizing the Hierarchical B-Spline basis, see [97], recovers this property.

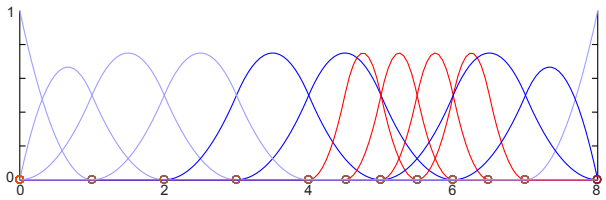


Fig. 1.9: Quadratic hierarchical B-Spline basis function. Coarse B-Spline functions are colored in blue and finer ones in red.

The basis function substitution is done without modifying the nearest neighbour coarse functions. It leaves some elements with more than $p+1$ non-zero functions on their support, what is called a overloaded element. The concept of overloaded element or overloaded knot interval was introduced by Dokken et al. [37]. It defines the overlap, respectively by more than $(p+1)$, $(p+1)(q+1)$ or $(p+1)(q+1)(r+1)$ basic functions by element or knot interval for a 1D, 2D or 3D basis. This happens because the large support of the coarse basis functions overlap with the support of several fine-scale ones. On Figure 1.10, the overloaded element are colored in light red and the too wide functions are colored in blue for the coarse level and red for the finer one.

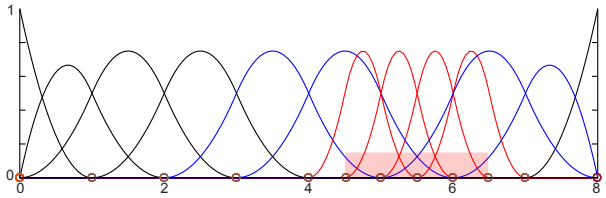


Fig. 1.10: Quadratic hierarchical B-Spline basis function. Coarse B-Spline functions are colored in black and blue and finer ones in red. The overlap of more than $p+1$ functions from knots 4.5 to 6.5 is highlighted in red.

1.3 Truncated Hierarchical B-Splines

The Hierarchical B-Splines allow for local refinement and are easy to define, but the number of overlapping basis functions can increase very rapidly with the introduction of new levels. This behaviour has a negative impact on the solution of linear system of algebraic equations associated to the solution of the discrete variational problem. From a computational point of view, a higher number of overlaps means that we add more elements in the system matrices and perform more functions evaluations. This increases the assembly time required to build such matrices as well as their sparsity.

Truncated Hierarchical B-Splines were introduced by Giannelli et al. [45], Kiss et al. [66], Giannelli et al. [46] to address these issues. Truncating the coarse basis functions reduces their support over refined area applying a scaling factor called truncature on the coarse functions that are in the neighbourood of the refined area, see Figure 1.11. The reader should refer to [45, 66, 46] for more details.

- Initialization : $T^0 = H^0$
- Recursive case : $T^{l+1} = T_A^{l+1} \cup T_B^{l+1}$ for $l = 0, ..., M-1$, where
  $T_A^{l+1} = \{trunc^{l+1} T : T \in T^l : supp\, T \not\subseteq \Omega^{l+1}\}$ (1.17)
  $T_B^{l+1} = H_B^{l+1}$
- $T = T^M$

The truncation part of the basis is performed from:

$$trunc^{l+1} T = T - \sum_{\substack{j : N_j \in N^{l+1} \\ supp N_j \subseteq \Omega^{l+1}}} \alpha_j N_j. \tag{1.18}$$

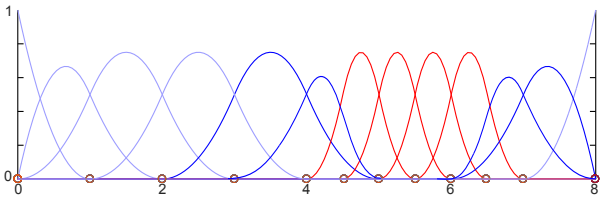


Fig. 1.11: Quadratic Truncated Hierarchical B-Spline basis functions. Coarse B-Spline functions are colored in blue and finer ones in red.

The Truncated Hierarchical basis naturally inherits the properties of the Classical Hierarchical basis, and also adds new ones:

- The functions in T are linearly independent.
- The spaces are nested.
- The basis maintains partition of unity.
- The cardinality of the basis is the same: $|H| = |T|$, considering the Hierarchical basis H defined on the same mesh as T,.
- The spaces spanned are the same: $spanH = spanT$.

Truncating the coarse basis functions reduces their support over the refined area compared to their counterparts and signicantly decreases the number of non-needed overlaps. However, it does not remove all the overloading, caused by the too large extent of the intermediate coarse basis functions on each side of the refinement area, dotted in Figure 1.12.

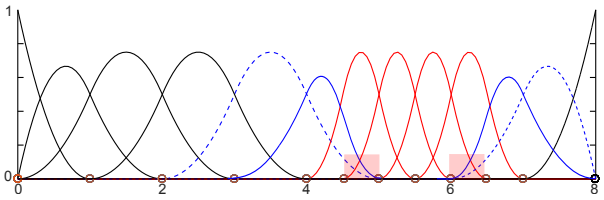


Fig. 1.12: Quadratic Truncated Hierarchical B-Spline basis functions. Coarse B-Spline functions are colored in black and blue and finer ones in red. The overlap area was reduced to knots from 4.5 to 5 and from 6 to 6.5: The dotted coarse function is still to extended.

Hierarchical strategies Another approach which shares some similarities with THB-Splines named Polynomial Splines over hierarchical T-meshes (PHT-Splines) was also introduced by Deng et al. [34], Nguyen-Thanh et al. [80], Wang et al. [102]. For conciseness, they will not be detailled here, the interested reader can refer to the previously cited references. These approaches build several B-Spline basis functions on several levels, substituting coarse functions by a set of refined ones but not checking the number of new basis functions introduced in the existing space.

1.4 Locally-Refined B-Splines

LR B-Splines were introduced by Dokken [36] and were applied to IGA by Johannessen et al. [57, 58]. This refinement method does not use a subdivision procedure like Hierarchical B-Splines but a local knot insertion, inserting one knot at a time and splitting old B-Spline basis functions into two new ones, generating few new functions, see Figure 1.13. Each refinement using the knot insertion creates two new functions in one direction.

The algorithm is similar to the classical knot insertion one in a B-Spline standard tensor-product, see [28] for details. Taking a new knot $\hat{\xi}$ to be inserted between the knots ξ_{i-1} et ξ_i in a knot vector Ξ for a 1D case, Ξ will be declined in two knot vectors Ξ_1 and Ξ_2, with an equivalent length, depending on the B-Spline degree:

$$\begin{aligned}\Xi &= [\xi_1, \xi_2, ..., \xi_{i-1}, \ \xi_i, ..., \xi_{p+1}, \xi_{p+2}],\\ \Xi_1 &= [\xi_1, \xi_2, ..., \xi_{i-1}, \hat{\xi}, \xi_i, ..., \xi_{p+1} \quad\ ],\\ \Xi_2 &= [\quad \xi_2, ..., \xi_{i-1}, \hat{\xi}, \xi_i, ..., \xi_{p+1}, \xi_{p+2}].\end{aligned} \tag{1.19}$$

The enriched B-Spline function B_Ξ is then a linear combination of the two B-Spline functions B_{Ξ_1} and B_{Ξ_2}, see Equation (3.1):

$$B_\Xi(\xi) = \alpha_1 B_{\Xi_1}(\xi) + \alpha_2 B_{\Xi_2}(\xi) \tag{1.20}$$

where

$$\begin{aligned}\alpha_1 &= \begin{cases} \frac{\hat{\xi}-\xi_1}{\xi_{p+1}-\xi_1} & \text{if} \quad \xi_1 \le \hat{\xi} \le \xi_{p+1}, \\ 1 & \text{if} \quad \xi_{p+1} \le \hat{\xi} \le \xi_{p+2}, \end{cases}\\ \alpha_2 &= \begin{cases} 1 & \text{if} \quad \xi_1 \le \hat{\xi} \le \xi_2, \\ \frac{\xi_{p+2}-\hat{\xi}}{\xi_{p+2}-\xi_2} & \text{if} \quad \xi_2 \le \hat{\xi} \le \xi_{p+2}. \end{cases}\end{aligned} \tag{1.21}$$

The weights α_1 and α_2 are necessary to ensure the partition of unity. Inserting new knots one at a time in the knot vector, the Spline space is enriched, see Figure 1.13, and the geometry remains unchanged. LR B-Splines have several advantages:

- The partition of unity is guaranteed.
- The link between coarse and fine levels is not needed after the construction of refinement. The LR B-Spline basis functions are defined with their local knot vector independently of the other functions.
- By construction, there is no overloaded element in 1D, and the basis is linearly independent.

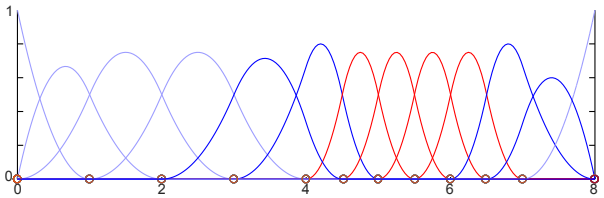


Fig. 1.13: Locally Refined B-Spline basis functions. Notice that there are no more than $(p+1)$ functions on each knot interval.

In the following, we will show that for 2D and 3D cases, the partition of unity is not guaranted but methods exist to ensure it. We will consider in this work that a sufficient condition to have a linearly independent LR B-Spline basis is to have no overloading, see [23, 24]. As will be explained later, the non-overloading is an important aspect in terms of the locally refined IGA implementation and its subsequent use in the data structures of a commercial FE solver.

1.4.1 Linear independence of LR B-Splines in 2D

Due to the shift of the ends of local knot vector, LR B-Spline refinement does not generate overloaded element in 1D. However, it is possible to have overloaded elements in 2D and 3D. This obtained basis functions are not guaranteed to be linearly independent for these cases. It happens for a bidirectionnal refinement of a rectangular area of elements in 2D or a tridirectionnal refinement for a cubic area of elements in 3D. These overloaded elements may cause the loss of linear independence and the basis functions may not be adapted for IGA.

Johannessen et al. [58, 57] introduced several diagrams for 2D refinement cases of rectangular area. Their purpose is to check several refinement schemes and to see if there is overloading or not.

1.4.1.1 Min Span

The min span refinement scheme, see Figure 1.14, consists in splitting at least one B-Spline but with the smallest possible footprint in the element. It means the split B-Spline is the one which has the smallest support on the element to be refined. If all the functions have the same support, the resulting refinement will not be unique.

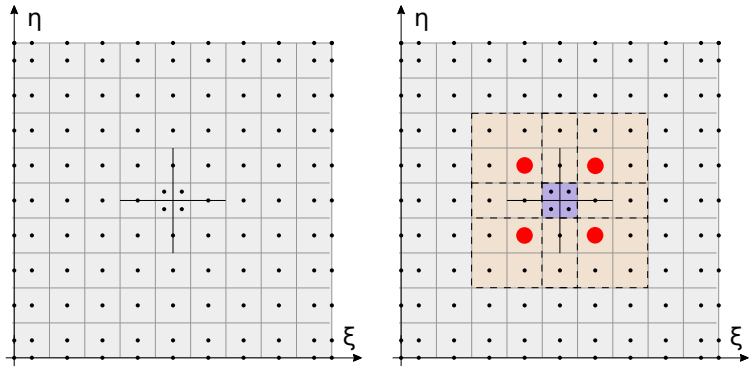


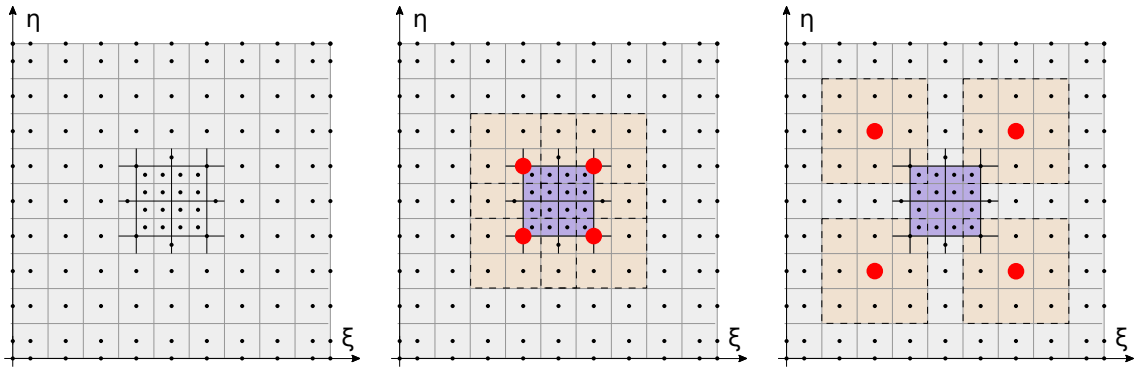
Fig. 1.14: Min span scheme in a 2D quadratic case. Overloaded elements are colored in blue. Too many B-Spline functions are non-zero on the elements. Some of them are dotted in red and their extent is colored in orange.

This scheme generates overloading, which is due to the functions that are in the diagonals of the refined area and which are illustrated in Figure 1.15(b) and Figure 1.15(c). The overloaded elements are colored in blue and the support for two of the extended functions is colored in orange in Figure 1.15. The extent of these functions were not modified and are too large. This scheme of refinement thus leads

to overloaded element in 2D. In 3D, it gives a number of overloaded elements all the more important.

1.4.1.2 Structured Mesh

Refinement according to the structured mesh method, see Figure 1.15(a), is described as a bi-directional refinement of all B-Spline functions fully contained in the rectangular refinement area. This scheme generates overloading, due to the functions that are in the diagonals of the refined area and which are illustrated in Figure 1.15(b) and Figure 1.15(c). The overloaded elements are colored in blue and the support for two of the extended functions is colored in orange in Figure 1.15. The extent of these functions were not modified and are too large. This refinement scheme thus leads to overloaded element in 2D and 3D.

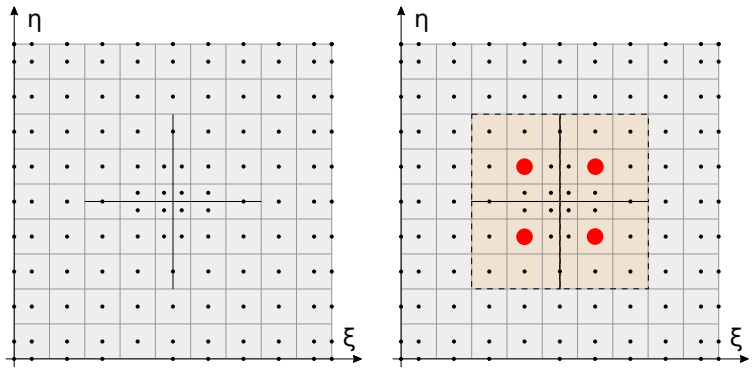


(a) (b) (c)

Fig. 1.15: Refinement according to structured mesh scheme for a 2D quadratic basis. Some elements, colored in blue, are overlodaded. There is still too many B-Spline functions on the elements. Some of them are dotted in red and their extent are colored in orange.

1.4.1.3 Full Span

The full span refinement scheme, see Figure 1.16(a), consists in extending the lines of refinement sufficiently so that they cut all the functions which are non-zero on the element to be refined. These additional cuts reduce a maximum the support of the functions which are rather outside the area of the refined elements. These external functions therefore no longer cause overloaded elements, see Figure 1.16(b). Non overloading is ensured for 2D and 3D refinements that are rectangular or cubic and have only outgoing corners.



(a) (b)

Fig. 1.16: Full span refinement scheme for a 2D quadratic basis: extension of refinement lines on p elements. There is no more overloaded element by orange basis functions.

The full span scheme is sufficient for most linear and cubic shape functions refinement cases [57]. This is also valid for 3D mesh refinements that have outgoing corners.

1.4.2 Linear independence checking and recovery methods

There are several practical methods, introduced by Johannessen et al. [57], for linear independence checking and recovery. These methods can ensure a priori the linear independence of a LR B-Spline basis with an empirical approach called the hand-in-hand principle. One can also recover it a posteriori with the peeling algorithm, or test the rank of the matrix which gives the relations between old and refined B-Splines like the tensor expansion to finally know if the resulting basis is linearly independent or not.

We will here briefly describe the first one, as it can be seen as a way to check for linear independence in a finte element code. For more details about the hand-in-hand principle and the tensor expansion, the reader can refer to the work of Dokken et al. [37].

Peeling algorithm The peeling algorithm is a linear independence recovery method, based on an a posteriori overloading elimination. It will deactivate several basis functions on overloaded element to recover the linear independence.

The algorithm is a two parts algorithm, see Algorithm (1). The first part tests the linear independence of a resulting mesh. A linear combination is simply found

if there is one or more overloaded element. The second part tries to remove some B-Splines in the mesh in the case of a linear combination. If a B-Spline function is non-zero only on one element, the function is removed. This procedure decreases the number of B-Splines on the element and is done while linear combinations are found on elements. When there is the correct number of functions depending on the dimension and the degree, the basis is proven linearly independent. If it is not possible, i.e., if it stays at least one function that can not be peeled, there may still exist a linear dependence. In that case, the peeling algorithm is not sufficient and other methods are required.

Algorithm 1 Peeling algorithm, from [57]

parameters: S {Spline space}; Ω {Parametric domain}; S_{LD} {Possible linearly dependent B-Splines}; Ω_{LD} {Areas of possible linear dependence}

$S_{LD} \leftarrow S$
$\Omega_{LD} \leftarrow S$
for every element $R \in \Omega$ **do**
    if R is locally linearly independent **then**
        $\Omega_{LD} \leftarrow \Omega_{LD} \setminus R$
        for every B-Spline B_i with support on R **do**
            $S_{LD} \leftarrow S_{LD} \setminus B_i$
        end for
    end if
end for
while Ω_{LD} or S_{LD} changed **do**
    for all elements $R \in \Omega_{LD}$ **do**
        if R has support of exactly one B-Spline $B_i \in S_{LD}$ **then**
            $\Omega_{LD} \leftarrow \Omega_{LD} \setminus R$
            $S_{LD} \leftarrow S_{LD} \setminus B_i$
        end if
    end for
end while

The peeling algorithm is a quick way to analyse the resulting mesh. Its definition is quite simple and can be applied for all dimensions. It is not sufficient to correct all linearly dependent meshes but it can prove the linear independence of a LR B-Spline basis. In that way, the peeling algorithm is a good a posteriori method.

1.5 T-Splines

T-Splines were proposed by Sederberg et al. [95] then used in IGA by Bazilevs et al. [8]. One of the principal differences lies on the fact that T-Spline basis functions are defined on local knot vectors. T-Splines allow to build meshes where the global tensor product is not a feature and where local refinement goes through T-junctions, analogous to hanging nodes for classical FEA.

Several T-Splines versions are available including T-Splines [95, 96], analysis-suitable T-Splines [94], modified T-Splines [64], weighted T-Splines [71, 70], hierarchical analysis-suitable T-Splines [41], or truncated T-Splines [104]. Each approach corrects the shortcomings of the previous one, like the lack of linear independence or partition of unity.

1.6 Isogeometric contact

Since its introduction, IGA showed a high potential compared to FEA to represent exact complex geometries, through the higher order and higher continuity that bring B-Spline and NURBS basis functions. The description of interacting surfaces with large displacements and large sliding, the non-linearity, the non-smoothness or the potential ill-conditioning which are still research challenges in FEA can be improved with IGA. However, the basis functions estimation could add more complexity and heavy computational costs to contact detection. This challenge is a key-point on the fate of IGA in computational software.

Several IGA contact formulations have been developed in the last years. These formulations derive for the most from existing contact treatments and algorithms developped for FEA. It takes advantages like the robustness, the efficiency and the ease of implementation of FEA contact methods besides benefits from the specific properties of IGA. But it also shares the disadvantages of FEA contact formulations like the contact surface discretization.

In the following, contact formulations and contact surface discretization approaches in IGA will be reviewed. These methods have been applied to implement the contact conditions at some discrete nodes as specific collocation points or to treat the contact constraints continuously.

1.6.1 Penalty methods

The penalty method is commonly used to solve algorithms that do not fulfill the inf-sup or Ladyzhenskaya–Babuška–Brezzi stability condition. It introduces a force

at the contact detection points that has penetrated across the target surface with the express purpose of eliminating the penetration:

$$F_c = k_c \times D_p, \tag{1.22}$$

for contact detection points that penetrate across the target, and

$$F_c = 0. \tag{1.23}$$

Otherwise, for open contact detection points. k_c is the contact stiffness, also called penalty stiffness. It is a predetermined property of the contact element. D_p is the penetration at the contact element.

Hence, the larger the penetration, the greater the calculated force. The challenge here is that the magnitude of the force necessary to prevent penetration is completely unknown beforehand. Obviously, the force needs to be large enough to push the contact surface back to the target surface and, thereby, eliminate unwanted penetration. But not so large that it pushes the contact completely off the target surface, causing error and instability. It is important that the contact stiffness be large enough so that the resulting penetration is negligibly small, but the contact stiffness can not be so large as to cause instability and nonconvergence. This same strategy is used both in the opposite direction to prevent separation (with bonded or no-separation behaviors) and in the tangential direction to enforce frictional resistance and no-sliding behavior.

A two-pass version of penalty called the Gauss-point-to-segment "two half-pass formulation" (GPTS-2hp) was introduced by Lu [73]. The contact integral is computed twice, switching the role of slave and master surfaces.

A positive aspect of the penalty method is that it is simple. The drawbacks of penalty-based formulations are unphysical penetrations and bad conditioning of the system of equations. Of course, this penetration is necessary for a contact force to be generated. Despite that, this formulation is often used in IGA as part of its integration into a finite element software.

Generally, the control polygon does not interpolate the physical geometry and so can not be used in a similar way that nodes are used for contact surface discretization in the classical node-to-surface algorithm on FEA. Collocation points which lie on the physical surface have to be defined and used. The location of these points and where to collocate the contact integrals is not trivial as in FEA. Several posibilities have been introduced:

1.6.1.1 Knot to surface

Temizer et al. [98] and Lu [73] introduced the first node to segment approach using collocation points as knot entries of the parameterization, at the vertices of the Bézier elements. However, this surface discretization provides generally fewer points than the number of control points. The contact formulation would be underconstrained as the number of contact constraints is less than the number of degrees of freedom associated to the surface.

1.6.1.2 Gauss point to segment

The Gauss point to segment formulation was introduced by Temizer et al. [98] and Lu [73] for 2D and 3D frictionless contact. The collocation points and the contact constraints enforcement do not take place at the physical location of the knot vector entries, but at the predetermined physical location of Gauss-Legendre quadrature points. Since contact force values are known directly at the integration points, their contribution to the element can directly be integrated.

The GPTS approach was extended to the 2D frictional setting in [30]. Temizer et al. [98] formulated and tested the extension to thermomechanical contact. More recently, the GPTS algorithm was adopted in a 2D and 3D frictionless setting by Dimitri et al. [35] in combination with T-Spline basis functions to demonstrate the advantages of local refinement. As its FEA counterpart, the GPTS formulation in the isogeometric setting is characterized by the simplicity of formulation and implementation and the possibility to obtain qualitatively good results for very coarse meshes.

1.6.1.3 Greville, Demko and Botella collocation points

Another way to collocate points on the surface is to set physical points in one-to-one correspondence with the control points associated to the surface. Such sets are, e.g., the Greville, Demko or Botella abscissae, and were introduced by Matzen et al. [75], Matzen and Bischoff [74]. Greville or Botella points enforce the contact and their locations are obtained straigthforwardly and Demko points are obtained by computing a complex iterative algorithm.

1.6.1.4 Bilinear quadrilateral collocation points

The three last interpolation methods use the knot vector characteristics, via its discrete values or via particular points position. Benson et al. [13, 14] introduced another way to set collocation points by considering the discretization at each element level and no longer the patch level. The collocation points are thus set on

the surface to construct a bilinear quadrilateral interpolation of each element of the contact surface. Spline element are fitted by a set at interpolation points depending on the desired accuracy in the solution of the contact problem.

1.6.1.5 Using the real NURBS or B-Spline surface

While the slave surface is always approximated with interpolation elements, the master surface may either be approximated or may use the real B-Spline or NURBS contact surface. It was introduced by Benson et al. [13, 14] for the case of a bilinear quadrilateral collocation approach of the slave surface where a nonlinear closest-point projection procedure is used to compute contact candidate area. This procedure brings precision but its computationnal cost is very important. It is possible to use it only on particular cases where the quality of the contact representation has to be improved.

1.6.2 Mortar methods

The mortar method is a segment-to-segment strategy, first introduced for contact analysis by Belgacem et al. [9]. In this method, one side of the domain is denoted as mortar one and the other as nonmortar. Usually, the surface with more control points than the other is chosen as the nonmortar one.

Integration points of nonmortar surface are used as slave nodes and are projected onto the mortar surface in the physical domain. A point inversion technique is then used in the mortar domain to determine the position in the parametric space of the projection point. This matching technique for contact surfaces is performed for all the integration points on the real contact surface. The contact constraints are thus computed along the interface boundary in a weak sense. The mortar method is well known to satisfy the LBB condition.

Mortar methods deliver several advantages, like the quality of the local results and the contact pressures (e.g. in the classical Hertz problem). Mortar formulations applied to IGA were introduced by Temizer et al. [98] and by Kim and Youn [65] in the 2D frictionless setting, for 2D friction in [30]. It has been extended to 3D in [99] and [31] in the frictionless and frictional settings, respectively. More recently was introduced contact for hierarchical NURBS using Mortar formulation in [97]. Several Mortar formulations are given in [108].

The non-interpolatory nature of the Spline basis functions implies that the mortar contact constraints do not have the immediate physical meaning that they possess in the FEA setting. This however does not affect the consistency nor the performance of the algorithm. However, compared to the penalty method, the mortar-

based approaches and their algorithms are computationally expensive, due to the mortar integrals to be computed.

1.6.3 Lagrange multiplier methods

Lagrange multiplier methods is also one of the most common used approaches to enforce surface contact. Unlike penalty methods, the Lagrange multiplier method does not need a user defined parameter. Lagrange multipliers are introduced into the equation of motion and are related to the surface contact forces. The method proceeds by treating the Lagrange multipliers as an unknown of the equation. This methods also fulfills the LBB stability condition if the Lagrange multiplier space is properly chosen.

It allows as well to use several discretization approaches with their advantages and several contact representations as the node to segment or the segment to segment ones. Segment-to-segment formulations have some advantages in solving large deformation problems, since they avoid locking and overconstraining and provide optimal convergence rates. The contact constraints are not imposed pointwise at a finite number of slave nodes but are defined along the entire contact boundary and therefore a more complete coupling between the degrees of freedom of the contact surfaces is obtained.

The Lagrange multiplier method has been used in the IGA context by Lu [73]. It can also be combined with Mortar methods, see [31, 32, 75].

1.7 A brief history of IGA integration into FE codes

Cottrell et al. [28] well described the IGA and how to integrate it into conventional finite element codes, in a similar way that are implemented finite elements, see Figure 1.17. The green blocks indicate the differences from the conventional finite element calculation. The blue blocks are identical whether it is isogeometric analysis or finite element analysis.

A relative important part of the existing code can be used or slightly modified in order to integrate these new elements. The basis functions are of course different. The part of the code generating these functions will therefore have to be modified, considering the new variables specific to B-Splines and NURBS like knot vectors or control points weights.

Another difference concerning control points is their generally non interpolating property, unlike finite element nodes. The boundary condition application needs a

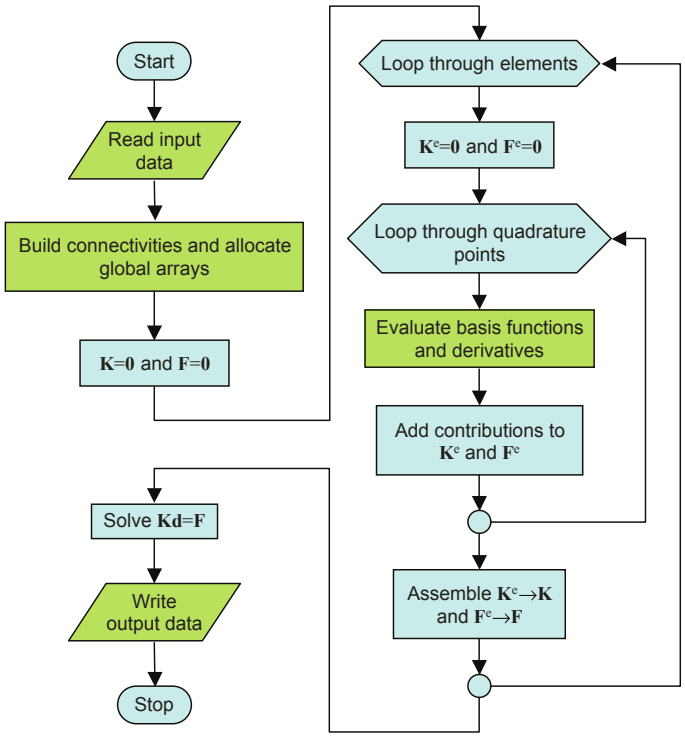


Fig. 1.17: B-Spline or NURBS calculation procedure in an implicite code, from [28].

specific attention according to this point.

A loop on the patches is needed in addition to the element loop, in case of multipatch models. The patch aspect has no equivalent in finite elements.The simpliest way to glue patches is to use conforming C^0 discretization on patch boundaries. The C^0 stitching is identical to the C^0 stitching used in finite elements.

Finally, like in any finite element code, in order to transfer control points contribution from the global numbering to the local numbering, a connectivity table is needed.

1.7.1 Bézier extraction

The Bézier extraction, introduced by Borden et al. [18], maps a piecewise Bernstein polynomial basis into a B-Spline or T-Spline basis and allows to define C^0 Bézier elements as finite element.

The following explanations are given for a B-Spline basis. The starting point is

the knot vector associated to the basis, e.g.:

$$\Xi = \{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}. \tag{1.24}$$

The associated B-Spline basis, control points and B-Spline curve are given in Figure 1.18(a). Several knot insertions of internal knot values are then performed in order to increase the multiplicity of each of them to p. The set of knot to be inserted in Equation (1.24) are given as follows:

$$\overline{\Xi} = \{1, 1, 2, 2, 3, 3\}. \tag{1.25}$$

Doing that insertion, the B-Spline elements defined by the knot spans are converted into a set of C^0 Bézier elements. The corresponding control point, basis functions and curve are given in Figure 1.18(b). Notice that the obtained curve is not changed.

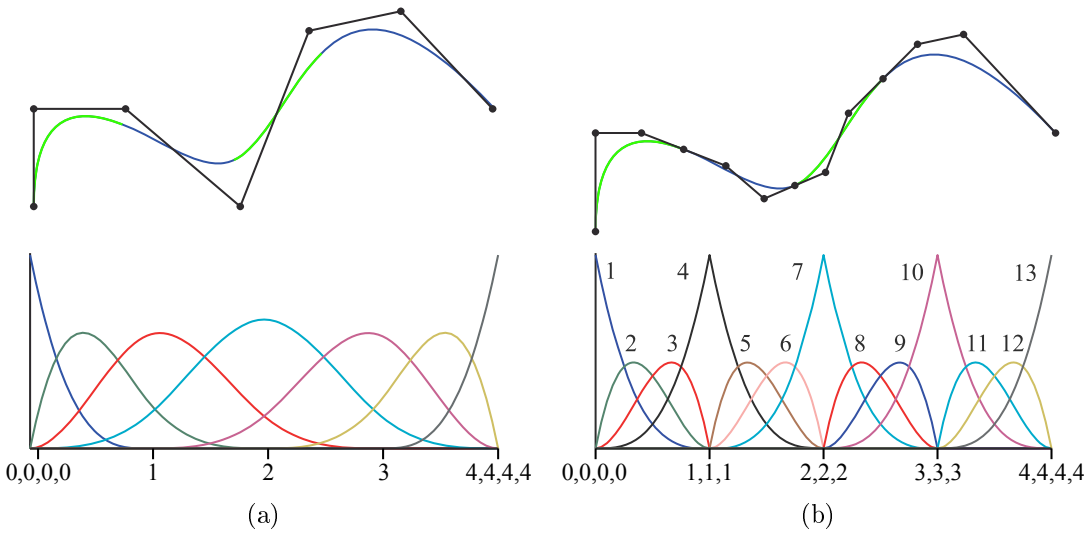


Fig. 1.18: Insertion of knots in the knot vector to obtain C^0 Bézier elements. a) Initial B-Spline basis and curve. b) Bernstein basis and curve. From [18].

The Bézier extraction operator is constructed combining the knot insertion coefficient into a matrix C and gives a relation between the B-Spline basis $B(\xi)$ and the Bernstein one $N(\xi)$, as:

$$N(\xi) = CB(\xi). \tag{1.26}$$

For more details about matrix C construction, we refer to the work of Borden et al. [18]. The integration of IGA with this technique can be done without modifying an important part of an existing finite element code. Bézier elements are given to

the solver in the same way that finite elements are. The knot vectors are the only data needed to compute the Bézier extraction operator. It can be computed in the preprocessor and directly given in the data file. In this way, the development effort is minimized.

1.7.2 IGA experiments in research codes

In the last years, IGA libraries were introduced in the academia, trying to constitute teaching and prototyping tools. GeoPDEs was the first open source library dedicated for the research on IGA, written in Octave and Matlab, see [29]. Others were then introduced in C++ like G+Smo library [60] and Igatools [85].

Several research software were also developed, like FEAP, a finite element analysis program which includes an isogeometric analysis library FEAP IsoGeometric. OOFEM is another finite element code for multi-physics with object oriented architecture, also in open source [90]. MIGFEM can also be cited as an IGA code implemented in Matlab which can support 2D and 3D fracture, see [79]. Others developments can also be found in Java and give the basic framework of a main object implementation [109].

In the LaMCoS, IGA have been treated for a few years. Indeed different aspects were discussed, like the shear and membrane locking in thick plates and shells [19], the solid shell element [21, 20, 22], the shape optimization of shell structures [52], the generation of NURBS volumetric mesh [3, 4, 6], until the integration in an industrial code of explicit dynamic computation [38].

Others finite element research codes and libraries are more oriented on the computational aspect of IGA like PetIGA [26], an open source framework for high performance isogeometric analysis heavily based on PETSc, using the C language. From the solver side, we find IGA-ADS as a solver which enables parallel multi-core simulations [72].

1.7.3 IGA implementation in industrial software

1.7.3.1 Abaqus

IGA was implemented for B-Spline and NURBS in a native way in the implicit solver Abaqus standard software via user subroutines, see [3, 38]. A new type of user element was defined. The .inp input data file was used to define the mesh as a classical finite element mesh, i.e. with the connectivity, the control points coordinates which are assimiled as nodes, the integration rules, etc... Additionnal data like the weight associated to each control point or the knot vectors are given in an additionnal .nb data file. The computation is done inside the software. A least

square projection is used to post-process the results on a finite element projection mesh with an Abaqus internal command.

Another implementation for T-Splines in Abaqus was done by Liu et al. [69] and Lai et al. [67] using the Bézier extraction. A Rhino plugin generates an Abaqus .inp file containing in the same way the données usuelles aux FE for the T-Splines elements. A second second .bezier file which contains the weigths, and the Bézier extractor.

1.7.3.2 LS-Dyna

LS-Dyna introduced IGA in a native way to implement tridimensional NURBS. The Bézier extraction was also implemented, especially for the use of T-Splines. IGA simulations were firstly done for shells [14, 15] and for blended shells [16]. The simulations demonstrated the quality of obtained results and their accuracy. Attempts to implement solid formulation have also been done [17]. Trimming surface were given in the Spline context in [76]. The mass scaling and the time increment estimates were also treated in [17].

1.7.4 The explicit solver Radioss

Radioss [106] is an explicit finite element solver for crash and impact simulation developped by Altair. It can model and solve complex non-linear behavior with a multitude of material and failure models. Radioss has built-in multi-physics capabilities for fluid-structure interaction and blast simulation. Limited implicit solver capabilities support gravity equilibrium as well as spring back analyses.

It has been used for crashworthiness studies in the automotive industry for over 25 years. A complete library of crash test dummy models is available for full vehicle crash simulations. Radioss has been widely used in defense applications and in the electronics industry for the verification of designs.

Applications of Radioss include automotive crash, passenger safety in cars, buses and trains, electronics device drop test, sheet metal stamping, blast impact, bird strike, airplane ditching or landing on water, bullet penetration of armor, etc.

Altair is interested in the integration of IGA in the explicit code Radioss.

Implementation and use within the Radioss Solver

Contents

2.1 Introduction

Since Radioss does not manage the B-Spline and NURBS functions in the input files, nor in the solver part and the display of curved elements, it is a complete job to implement the IGA.

In this chapter will be shown how IGA was implemented in Radioss for tridimentionnal cases. The implementation has been done in Fortran 95. The notions of parametric space, polynomial degree by direction, weights or the recursive nature of the basis functions are quite new and are more complicated compared to the existing classical finite element implementation. However, the implementation of IGA is more general because of these features and will be more easily adaptable to different geometries and degrees of the basis functions.

2.1.1 Data structures needed for FEA and IGA

Traditionnal FEA codes have an input data file containing the nodes coordinates, the element type with its associated numerical integration rule, and the connectivity which connects an element with its nodes. Finite element codes using isogeometric analysis require some additional input data compared to traditionnal ones: control points weights, knots vectors, polynomial degrees and a different element connectivity, see Table 2.1.

FEA	IGA using B-Splines and NURBS
Nodal coordinates	Control points coordinates
Element type	Polynomial degree
Table of connectivity	Element connectivity
Numerical integration rule	Numerical integration rule
	Control points weights
	Knot vectors

Table 2.1: Data needed for FEA and IGA.

Control points are counterparts of nodes and have associated weights. Their coordinates are stored as nodal ones and weights are given by adding real variables. Knots vectors which describe each of the patches are also passed through the input file in a specific format created for the isogeometric elements, giving in the same time the polynomial degree for each parametric direction. The element connectivity is also different. The element number is stored with associated shape function indexes. The position of these functions with respect to the global shape function index is also gathered. The detailed definitions and constructions of the element

connectivity are given by Cottrell et al. [28].

2.1.2 Global structure of Radioss and analysis procedure

In a similar way to the flowchart given for classical implicit finite element code by Cottrell et al. [28], it can be defined for explicit code to highlight the specific existing routines that have to be replaced in accordance with IGA specificities, see Figure 2.1.

B-Spline or NURBS basis functions are evaluated in the routine also given in [28] and adapted to Fortran. The specificity of explicit simulation compared to implicit one is the need to compute at each time step a time increment which will ensure the stability of the simulation. The geometric specificities of the B-Spline and NURBS geometries, the span and the smooth characteristic of their basis functions induce some modificitations of existing time increment estimation methods. Finally, output data file has to be written more than once, depending on the sorting frequency. The format of that output will be specific to the isogeometric basis and is adapted to existing postprocessing software.

2.1.3 Equilibrium equation and matrix/vector calculation

To go into more detail about the implementation and calculation of the different elements needed for explicit resolution, we can begin from the dynamic equilibrium equation of a system, written at any time step t:

$$\underline{\underline{M}}\underline{\gamma} + \underline{\underline{C}}\underline{v} + \underline{\underline{K}}\underline{u} = \underline{F_{ext}}, \tag{2.1}$$

with $\underline{\underline{M}}$ the mass matrix, $\underline{\underline{C}}$ the damping matrix, $\underline{\underline{K}}$ the stiffness matrix. $\underline{u}$, $\underline{v}$ and $\underline{\gamma}$ are respectively the displacement, the velocity and the acceleration vectors. $\underline{F_{ext}}$ is the external forces vector. This vector mainly includes applied forces, gravity, contact forces. In general, in explicit solvers, the damping effects can be ignored and the stiffness matrix $\underline{\underline{K}}$ is never directly assembled. Equation (2.1) is then modified using the internal forces vector $\underline{F_{int}}$:

$$\underline{\underline{M}}\underline{\gamma} = \underline{F_{ext}} - \underline{F_{int}}. \tag{2.2}$$

Note that in the present works, we will use only 3D solid models. It will not therefore be necessary to compute rotations and moments, as could be the case for shells.

The implementation of isogeometric elements in Radioss is done in two parts. A first part for calculating the diagonal mass matrix, the inertia and the critical time increment at the initial time step t_0, and a second part for the calculation of the

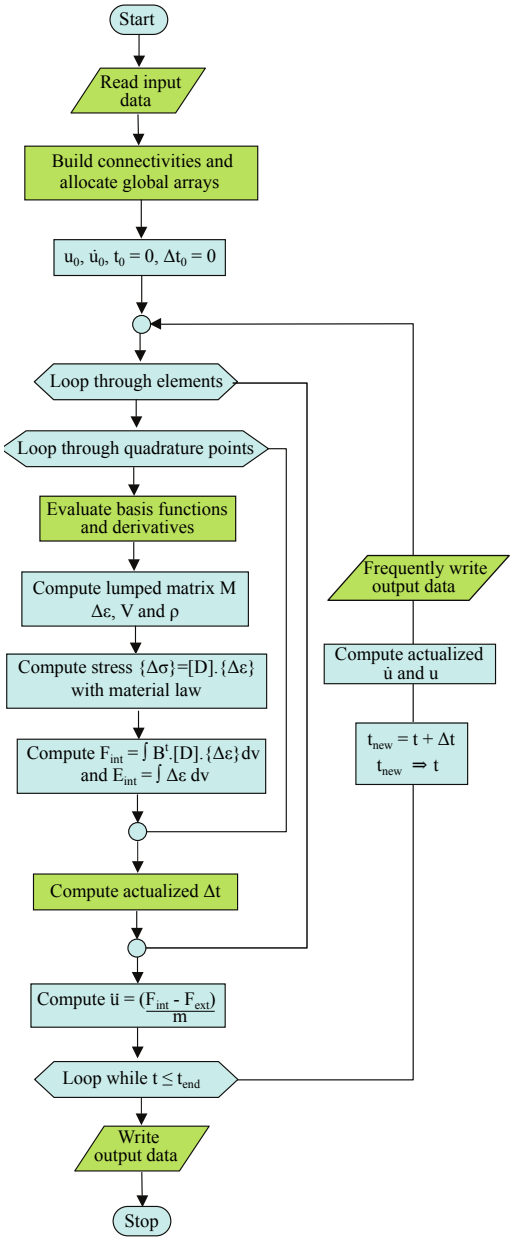


Fig. 2.1: Flowchart of a classical explicit finite element code, converted for IGA by replacing the routines shown in green by specific IGA code.

strain increments, the stresses, the force assembly and updating the critical time increment. The routines contained in these two parts are plugged into the Radioss source code.

2.1.3.1 Discretization: Central difference

The central difference resolution method is used in Radioss. Equation (2.1) and Equation (2.2) are second order equations that can be solved using several discretization methods. Implicit analysis approaches solve the problems given in Equation (2.1) at a given time step t using unconditionally stable integration method. This method is a part of the Newmark family scheme [54], where the two parameters $\alpha = 1/2$ and $\beta = 1/4$. Explicit methods are other methods for which the displacements at a given time step t explicitly depend on the variables at the previous instant. This is the case of the central difference method [89], where the two parameters α and β are respectively equal to 1/2 and 0. This resolution method is commonly used in explicit codes and in particular in Radioss. It thus makes it possible by knowing the state of a system at the instant t^n and $t^{(n-1)}$, to compute its state at the instant $t^{(n+1)}$.

Given the coordinates at time steps t^n and $t^{(n-1)}$, we can calculate the velocity $\underline{v}(t^{(n-\frac{1}{2})})$:

$$\underline{v}(t^{n-\frac{1}{2}}) = \frac{\underline{x}(t^n) - \underline{x}(t^{n-1})}{t^n - t^{n-1}}. \tag{2.3}$$

The equilibrium equation at time step t^n allows to calculate $\underline{\gamma}$, and then compute $\underline{v}(t^{n+\frac{1}{2}})$ and $\underline{x}(t^{n+1})$ as follows:

$$\underline{v}(t^{n+\frac{1}{2}}) = \underline{v}(t^{n-\frac{1}{2}}) + (t^{n+1} - t^n) \times \underline{\gamma}(t^n), \tag{2.4}$$

$$\underline{x}(t^{n+1}) = \underline{x}(t^n) + (t^{n+1} - t^n) \times \underline{v}(t^{n+\frac{1}{2}}). \tag{2.5}$$

With the new coordinates at time step t^{n+1}, a volume and a new density are computed thanks to the mass balance equation. The diagonal mass matrix, the internal forces vector $\underline{F_{int}}$ and the external forces vector $\underline{F_{ext}}$ are thus used to write the equilibrium equation, which gives the acceleration at time step t^{n+1}.

In explicit codes, the time increment between two time steps is usually not constant. The writing of Equation (2.4) and Equation (2.5) is thus modified, with the old time increment dt_1 and the new dt_2:

$$\underline{v}(t^{n+\frac{dt_2}{2}}) = \underline{v}(t^{n-\frac{dt_1}{2}}) + (\frac{dt_1 + dt_2}{2}) \times \underline{\gamma}(t^n), \tag{2.6}$$

$$\underline{x}(t^{n+dt_2}) = \underline{x}(t^n) + dt_2 \times \underline{v}(t^{n+\frac{dt_2}{2}}). \tag{2.7}$$

This scheme is conditionally stable and non-dissipative, which means that the time increment is limited to ensure that the solution does not grow boundlessly. The stability condition [84] can be applied to a one degree-of-freedom of a system without

damping and the dynamic equilibrium equation at time t^n given in Equation (2.2) would defined the critical time increment as:

$$\Delta t \leq \frac{2}{\omega_{max}}, \tag{2.8}$$

with ω_{max} the maximum natural frequency of the system which ensure the following relation:

$$det(K - \omega^2 M) = 0. \tag{2.9}$$

Another consideration in the time integration stability concerns the type of problem which is analyzed. For example in the analysis of wave propagation, a large number of frequencies are excited in the system. That is not always the case of structural dynamic problems. In a wave shock propagation problem, the time increment must be small enough in order to excite all frequencies in the finite element mesh. This requires short time increment so that the shock wave does not miss any node when traveling through the mesh. It follows that the time increment should be limited by the following relation:

$$\Delta t \leq \frac{L_c}{c_0}. \tag{2.10}$$

L_c is the characteristic element length, representing the shortest road for a wave arriving on a node to cross the element, and c_0 is the speed of sound in the material.

2.2 3D solid isogeometric element formulation

One of the main goals of this work is to highlight the capabilities of IGA in a dynamic context comparing several isogeometric elements and their FEA counterparts. Thus we realized the implementation by following the FEA implementation pathway. The following sections will present the construction of several matrix, like mass matrix, the computation of stress, the estimation of critical time increment all allong the run, and the contact formulation modified in order to use that new kind of element.

2.2.1 Tridimensional isogeometric element

We consider a tridimentionnal B-Spline or NURBS element of polynomial degree p, q and r, composed of $(p+1)(q+1)(r+1)$ control points. Coordinates inside this element can be interpolated as:

$$\underline{x}(\xi, \eta, \zeta) = \sum_{i=1}^{nctrl} N_i(\xi, \eta, \zeta)\underline{x}_i, \tag{2.11}$$

with *nctrl* the number of control points of the element. Radioss uses an updated Lagrangian displacement-based formulation. The displacement field and incremental displacement field can be interpolated as follows:

$$\underline{u}(\xi,\eta,\zeta) = \sum_{i=1}^{nctrl} N_i(\xi,\eta,\zeta)\underline{u}_i, \tag{2.12}$$

$$\underline{\delta u}(\xi,\eta,\zeta) = \sum_{i=1}^{nctrl} N_i(\xi,\eta,\zeta)\underline{\delta u}_i. \tag{2.13}$$

Full quadrature The implementation of B-Spline or NURBS element was done for full quadrature, with $(p+1)(q+1)(r+1)$ integration points by element. The interested reader can refer to the large litterature on reduced quadrature for IGA [56, 93, 2, 50].

2.2.1.1 Mass matrix calculation

The mass matrix $\underline{\underline{M}}$ is the consistent mass matrix and is defined in Equation (2.14). The diagonal mass matrix, used in Radioss, is defined using a row summation, see Equation (2.15). This lumping method reduces computational time considerably as no matrix inversion is necessary.

$$M_{(a,b)} = \int_{\Omega_e} N_a \rho N_b d\Omega. \tag{2.14}$$

$$M_{(a,a)} = \sum_b \int_{\Omega_e} N_a \rho N_b d\Omega = \int_{\Omega_e} N_a \rho \left(\sum_b N_b\right) d\Omega. \tag{2.15}$$

Applying the partition of unity $\sum_b N_b = 1$, we obtain the diagonal terms:

$$M_{(a,a)} = \int_{\Omega_e} \rho N_a d\Omega = \int_x \int_y \int_z \rho N_a dx dy dz. \tag{2.16}$$

Jacobian mapping The mass matrix expression is given in the physical space (x,y,z), whereas shape functions N_a are known in parametric space (ξ,η,ζ), and the Gauss points are in the parent space $[-1;1]$ for each direction. The computation of the integral is transported in the parametric space, giving a new mass matrix expression:

$$M_{(a,a)} = \int_\zeta \int_\eta \int_\xi \rho N_a(\xi,\eta,\zeta) det(J) d\xi d\eta d\zeta, \tag{2.17}$$

with $det(J) = \frac{dxdydz}{d\xi d\eta d\zeta}$, then in the parent space:

$$M_{(a,a)} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \rho N_a(\xi,\eta,\zeta) det(J) det(J_{lin}) dudvdw, \tag{2.18}$$

with $det(J_{lin}) = \frac{d\xi d\eta d\zeta}{dudvdw}$.

By using numerical integration to discretize this integral, we finally obtain, with $w_{i,j,k}$ the Gauss points weights:

$$M_{(a,a)} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} \rho N_a(\xi_{i,j,k},\eta_{i,j,k},\zeta_{i,j,k}) det(J) det(J_{lin}) w_{i,j,k}. \tag{2.19}$$

2.2.1.2 Mass Conservation and update of ρ

The resolution of Equation (2.1) is done with an explicit scheme, that is to say that the state of a system at a given time t is explicitly defined by its state at the previous instant. This resolution method will be discussed in the following sections. The density is updated at each instant t from the density at time t_0 and from the two volumes V_0 and V from the relation:

$$\rho_0 V_0 = \rho V, \tag{2.20}$$

with the volume V definition:

$$V = \int_{\Omega_e} dv = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} det(J) det(J_{lin}) w_{i,j,k}. \tag{2.21}$$

The update of ρ is then obtained:

$$\rho = \frac{\rho_0 * \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} det(J_0) det(J_{lin_0}).w_{i,j,k}}{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} det(J) det(J_{lin_0}).w_{i,j,k}}. \tag{2.22}$$

This updated density will be used to update the mass matrix at each time t, then to calculate the strain increments and the internal forces.

2.2.1.3 Internal forces computation

The computation of forces vector and the internal energy uses a material law which can be formulated in strain or strain increment, depending the law.

Radioss material law Radioss material laws are usually written with the strain increments, i.e.:

$$\underline{\underline{\sigma}} = \underline{\underline{\sigma^-}} + \underline{\underline{D\Delta\varepsilon}}. \tag{2.23}$$

We have to calculate strain increments from displacements at the control points and the following relation:

$$\underline{\underline{\Delta\varepsilon}} = \frac{1}{2}(\underline{\underline{grad(\Delta u)}} + \underline{\underline{grad(\Delta u)^T}}) + \underline{\underline{\Delta\varepsilon^q}}, \tag{2.24}$$

with $\underline{\underline{\Delta\varepsilon^q}}$ the quadratic part of the gradient. In general, to have a simplified formulation, only the first part of this equation is used, i.e. small strains. The second part is often ignored, which gives for example for the xx strain:

$$\Delta\varepsilon_{xx} = \frac{\partial(v_x dt)}{\partial x} = \sum_i \frac{\partial N_i}{\partial x} v_{x_i} dt, \tag{2.25}$$

or for the xy one:

$$\Delta\varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial(v_y dt)}{\partial x} + \frac{\partial(v_x dt)}{\partial y}\right) = \frac{1}{2}\sum_i \left(\frac{\partial N_i}{\partial x} v_{y_i} + \frac{\partial N_i}{\partial y} v_{x_i}\right) dt. \tag{2.26}$$

These approximations are valid in many cases for small displacements. Since large displacements are possible in Radioss, the quadratic part of Equation (2.24) is added:

$$\Delta\varepsilon_{xx} = \frac{\partial(v_x dt)}{\partial x} + \left(\frac{\partial(v_x dt)}{\partial x}\right)^2 = \sum_i \frac{\partial N_i}{\partial x} v_{x_i} dt + \left(\sum_i \frac{\partial N_i}{\partial x} v_{x_i} dt\right)^2, \tag{2.27}$$

$$\begin{aligned}\Delta\varepsilon_{xy} &= \frac{1}{2}\left(\frac{\partial(v_y dt)}{\partial x} + \frac{\partial(v_x dt)}{\partial y} + \frac{\partial(v_y dt)}{\partial x}\frac{\partial(v_x dt)}{\partial y}\right) \\ &= \frac{1}{2}\left(\sum_i \left(\frac{\partial N_i}{\partial x} v_{y_i} + \frac{\partial N_i}{\partial y} v_{x_i}\right) dt + \sum_i \left(\frac{\partial N_i}{\partial x} v_{y_i} * \frac{\partial N_i}{\partial y} v_{x_i}\right) dt^2\right).\end{aligned} \tag{2.28}$$

Once the strain increments are computed, a material law is called and returns the stress $\underline{\underline{\sigma}}$ at the integration points and updates the internal energy using the updated volume between V_0 and V. These two last computation are given following:

$$\underline{F_{int}} = \int_{\Omega_e} \underline{\underline{B^t \sigma}}\, dv, \tag{2.29}$$

$$\in_{int} = \in_{int} + \int_{\Omega_e} \underline{\underline{\sigma}}.\underline{\underline{\Delta\varepsilon}}\, dv. \tag{2.30}$$

These internal energy and force are also computed by summing the contribution of each integration point to the control points, just as for the calculation of the mass matrix, with the Gauss points weight $w_{i,j,k}$.

2.2.2 3D geometries and models generation

Preprocessing is still a delicate step in IGA, especially for 3D models. Generating complex B-Spline or NURBS geometries can only be done using dedicated tools. Efforts have been made to develop robust and exploitable solutions. We can refer to the work of Al Akhras et al. [4, 6, 5] or Liu et al. [69], Wang and Qian [103], Xu et al. [107]. These methods bring the transition from CAD to analysis suitable isogeometric models, through 3D computer graphics and CAD software such as Rhinoceros. At the moment, there is no tool in HyperWorks that allows the preprocessing of B-Spline or NURBS geometries. Therefore, in the context of this study, a Rhinoceros plugin was used to define the isogeometric models and was modified to product analysis-suitable input files in the Radioss format, see the appendix B.

2.2.3 Post-processing

The purpose of the post-processing is to extract and display the results of the Radioss calculation. The problem for post-processing is the same as for pre-processing since the HyperWorks post-processing tool, HyperView, is only able to generate standard finite elements. If we import a B-Spline or a NURBS calculation result file as it is without any particular treatment, only the control points will be displayed since they will be considered as conventional nodes. A solution allowing the visualization of the results is to use a least square projection on a finite element mesh for each B-Spline or NURBS patch, see for example [40**?** , 78]. This projection mesh is defined so that each B-Spline or NURBS volume element is represented by 27 linear bricks. The fields defined on control points are directly interpolated with the basis functions. The ones defined at Gauss points are first projected using least squares. The scalar or vector fields are computed by patch constituting the Gram matrix. This matrix $\underline{\underline{M}}$ has an equal dimension to the number of control points of the complete patch. It is defined as follows:

$$M_{(a,b)} = \int_{\Omega} N_a(x) N_b(x)\ d\Omega = \sum_{n=1}^{nel} \sum_{k=1}^{ngauss} N_a(x_k) N_b(x_k) \omega_k. \tag{2.31}$$

For each result field i to be projected, a vector $\underline{R^i}$ is computed by:

$$R_a^i = \int_{\Omega} N_a(x) f^i(x)\ d\Omega = \sum_{n=1}^{nel} \sum_{k=1}^{ngauss} N_a(x_k) f^i(x_k) \omega_k. \tag{2.32}$$

The least squares problem is then solved on the set of each of the patches, for each result field i, by calculating the vector $\underline{F^i}$

$$\underline{F^i} = \underline{\underline{M}}^{-1}\underline{R^i}, \tag{2.33}$$

and projecting on the projection nodes j through the relation:

$$\underline{f^i}(x_j) = \sum_a N_a(x_j)\underline{F^i}. \tag{2.34}$$

This global least squares projection uses the L^2 norm and gives satisfactory results. However, this results in the resolution of a problem that is the same size as the original problem. Recently, Govindjee et al. [47] have introduced a local least squares method to obtain accurate results at a reasonable computing cost. In the case of this work, the Gram matrix that must be formed in order to perform the global least squares projection is identical to the mass matrix with a unit density on the same mesh.

This projection method does not benefit from the strengths of IGA. The represented surfaces are not continuous and the results are mostly extrapolated. This is the problem when a rich solution is projected on a degraded space. At the moment, it has been decided to project a 3D isogeometric element, regardless of its polynomial degree p, q and r, on a set of 27 linear bricks. This quality of discretization is fixed and will bring its limits for the representation of result fields with high gradients. However this solution of the 27 bricks, makes it possible to limit the size of the tables created solely for the display, even if the time spent in extracting the output results represents only a small part of the total cost of the simulation. A discretization dependent on the quality of the results, the polynomial degrees or the geometries of the initial model could be used and would allow a more faithful visualization. It would be advisable to define automatically the number of bricks on which the projection will be made by $(p+1)(q+1)(r+1)$, but in this case the size of the coordinate tables and constraints that would be built would be much higher. Using this post-processing method allow to display calculation results in HyperView.

2.2.4 Patch test: uniaxial traction

In order to validate to IGA implementation in the Radioss solver, a uniaxial tension test along the z axis on a 0.1 m side linear elastic cube is performed. This test consists in finding a uniform state of stress and strain for an unstructured assembly in tension. The test is considered as a quasi-static simulation and is presented without time increment estimate approach. Two symmetry planes following x and y direction are defined to model only one-eighth of the geometry. The modeled geometry is thus a cube of 0.05 m side, which is locked on its lower face in the z direction, on y on its right face and on x on the face of behind. This geometry will

Direction	Order and knot vectors for the linear patch	Order and knot vectors for the quadratic patch
ξ	p=1 $\quad \Xi = [0, 0, 1, 1]$	p=2 $\quad \Xi = [0, 0, 0, 1, 1, 1]$
η	q=1 $\quad H = [0, 0, 1, 1]$	q=2 $\quad H = [0, 0, 0, 1, 1, 1]$
ζ	r=1 $\quad Z = [0, 0, 1, 1]$	r=2 $\quad Z = [0, 0, 0, 1, 1, 1]$

Table 2.2: Knot vectors for both isogeometric element of the uniaxial traction test.

be constrained to a 0.005 m displacement along z on its upper face. The geometry and applied boundary conditions are given in Figure 2.2.

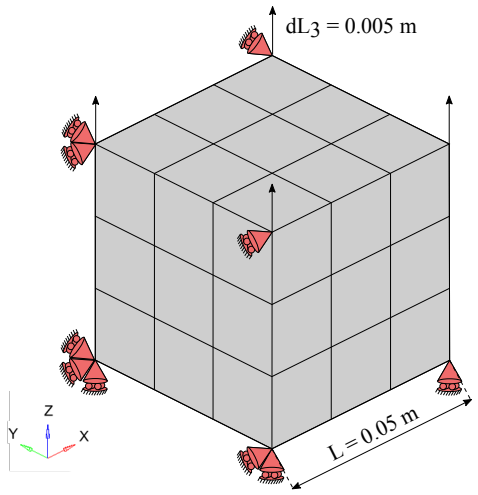


Fig. 2.2: Uniaxial traction: dimension, boundary conditions and mesh.

Two isogeometric meshes will be compared to their finite element counterparts. The 8-control point linear B-Spline element is compared to the 8-node linear brick (HA8 in the Radioss library). The quadratic B-Spline element with 27 control points is compared to the 20-node quadratic brick (S20 in the Radioss library). All these models are fully integrated, and the corresponding knot vectors for the two isogeometric elements are given in Table 2.2. The tensile behavior of a single element will be analyzed.

Analytic stress can be computed using the global volume. The theoretical calculation is made on the overall volume, i.e. a 0.1 m side cube in uniaxial tension with a displacement of 0.01 m. The elongation in the z direction is given as:

$$\varepsilon_{33} = ln\left(\frac{L_3 + dl_3}{L_3}\right) = \frac{0.1 + 0.01}{0.1} = 0.09531. \tag{2.35}$$

The two other strains ε_{33} and $\varepsilon_{11} = \varepsilon_{22}$ are given, with $\nu = 0.3$, by:

$$\varepsilon_{11} = -\varepsilon_{33}\nu = -0.09531 \times 0.3 = -0.028593. \tag{2.36}$$

With a Young modulus $E = 210\,000$ MPa, the value of the longitudinal stress in the z direction is 10 246 MPa, whereas the others components are zero. This longitudinal stress will serve as a reference.

2.2.4.1 Results

Table 2.3 shows the longitudinal stress for each mesh. The distribution of the displacements in the volume for x, y, and z is plotted on Figure 2.3.

	Error of longitudinal stress (%)
Radioss HA8	0.131
Linear B-Splines	0.131
Radioss S20	0.132
Quadratic B-Splines	0.132

Table 2.3: Uniaxial tensile test: relative error in longitudinal stress.

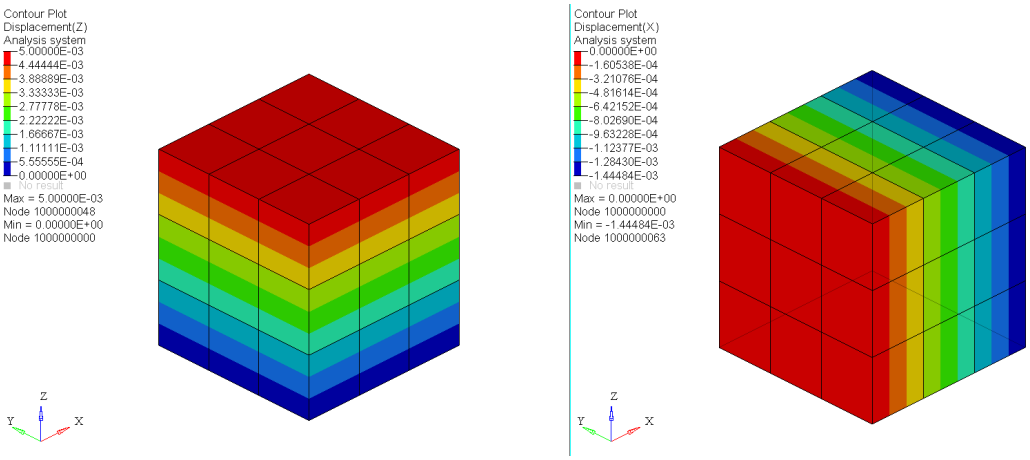


Fig. 2.3: x and z displacement, homogeneous in the brick section.

The displacements in the x and y directions are uniform for all models and the longitudinal stress is constant in the volume throughout the simulation. This behavior is the one that was expected and qualitatively validates this patch test. Relative errors on longitudinal stress, for both elements already existing in Radioss library or isogeometric elements, are minimal compared to the theoretical values (of the order of 0.1%). These values allow to quantitatively validate these models.

2.2.5 Distributed loads

The surface loadings distribution on the control points is generally given for any type of element as follows:

$$\underline{F} = \int_S p\underline{n}\, dS, \tag{2.37}$$

where p is the pressure value and $\underline{n}$ the normal vector computed depending on the element external face. The pressure is positively defined when it is facing inwards. The application of distributed loads specifically for isogeometric elements is quite simple to implement. It largely uses the shape functions and derivatives routine of [28]. Therefore, instead of computing the B-Spline basis functions in the 3D volume, they are computed on one external face and allow to have access to the normal vector and to the area of the outer face of the corresponding isogeometric element.

A definition of elements external faces has to be set. The B-Spline or NURBS patches parameterization gives a structured mesh for which it is easy to find an external face. For each outer face of an element, the normal vector is computed from the cross product of Jacobian-specific components $J_{ij} = \frac{dX_i}{dx_j}$, depending on the needed element face. A convention on the numbering of the external face is given in Table 2.4 and illustrated in Figure 2.4.

N_{face}	Corresponding parametrization	Normal vector	Parametric normal vector direction
Face 1	$\{(\xi,\eta,\zeta) : \xi = 0\}$	$J_{i2} \wedge J_{i3}$	ξ^+
Face 2	$\{(\xi,\eta,\zeta) : \xi = 1\}$	$J_{i3} \wedge J_{i2}$	ξ^-
Face 3	$\{(\xi,\eta,\zeta) : \eta = 0\}$	$J_{i3} \wedge J_{i1}$	η^+
Face 4	$\{(\xi,\eta,\zeta) : \eta = 1\}$	$J_{i1} \wedge J_{i3}$	η^-
Face 5	$\{(\xi,\eta,\zeta) : \zeta = 0\}$	$J_{i1} \wedge J_{i2}$	ζ^+
Face 6	$\{(\xi,\eta,\zeta) : \zeta = 1\}$	$J_{i2} \wedge J_{i1}$	ζ^-

Table 2.4: Parametric normal vector direction depending on the patch face.

The surface is then computed, e.g. for the face 1 with the normal direction ξ_- as follows:

$$S_{\xi_-} = \sum_{j=1}^{m}\sum_{k=1}^{o} \frac{dM_{j,q}(\eta_j)}{d\eta}\frac{dR_{k,r}(\zeta_k)}{d\zeta}\frac{d\eta d\zeta}{dydz} w_{\xi_-,\eta_j,\zeta_k}. \tag{2.38}$$

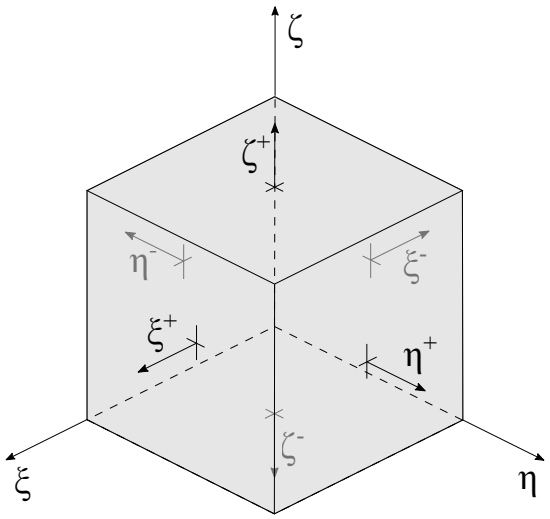


Fig. 2.4: Parametric normal vector direction.

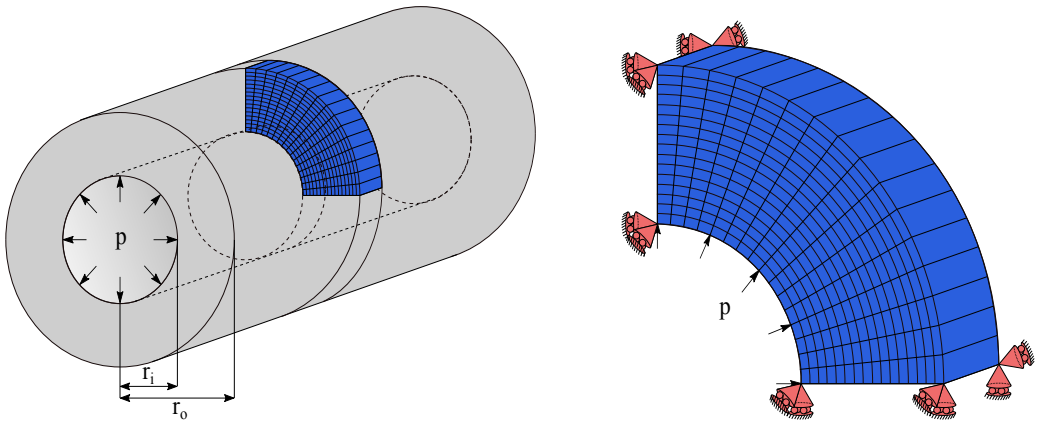
2.2.6 Internal pressure: Quarter disk

A thick-walled cylinder subjected to internal pressure is introduced to illustrate the distributed load computation. This example is a common illustration of IGA implementation, see [39, 61]. Analytical solutions of this problem are given in [33, 51]. A thick cylinder, with an inner radius $r_i = 100$ mm, an outer radius $r_o = 200$ mm is considered in 2D in plane strain. An internal pressure p is progressivly applied until it reaches the maximum value of $p_i = 0.14$ GPa. An elastic perfectly plastic material is used with a Young's Modulus of 120 GPa, a yield stress in pure tension of 0.24 GPa and a Poisson's ratio of 0.3.

Only a quarter of the cylinder is modeled, considering the inherent symmetry. The boundary condition and the loading is applied in consequence of this symmetry and the plane strain hypothesis. The defined mesh used in this example is composed of 64 C^1 quadratic NURBS elements, with only one element through the thickness of the cylinder. The geometry, the boundary conditions and the mesh used are given in Figure 2.5, and knot vectors with corresponding degrees are given in Table 2.5.

Direction	Order	Knot vector
ξ	p=2	$\Xi = \left[0, 0, 0, \frac{1}{16}, \frac{2}{16}, \ldots, \frac{14}{16}, \frac{15}{16}, 1, 1, 1\right]$
η	q=2	$H = \left[0, 0, 0, \frac{1}{16}, \frac{2}{16}, \ldots, \frac{14}{16}, \frac{15}{16}, 1, 1, 1\right]$
ζ	r=2	$Z = [0, 0, 0, 1, 1, 1]$

Table 2.5: Knot vectors for the quarter of the thick cylinder.



(a) (b)

Fig. 2.5: Thick cylinder under pressure. a) Dimensions, loading conditions and mesh used, b) mesh detail and corresponding boundary conditions.

Analytic expressions give the hoop and the radial stress in the cylinder in the elastic case:

$$\sigma_{rr} = \frac{p_c r_c^2}{r_o^2 - r_c^2}\left[1 - \frac{r_c^2}{r^2}\right] \; ; \sigma_{\theta\theta} = \frac{p_c r_c^2}{r_o^2 - r_c^2}\left[1 + \frac{r_c^2}{r^2}\right]. \tag{2.39}$$

For internal pressurization, the radial stress and circumferential stress are maximum at the inner radius. The Von Mises stress is given by the following formula:

$$\sigma_{VM} = p * \sqrt{\frac{3r_o^4 + r_i^4}{(r_o^2 - r_i^2)^2}} = \sqrt{\sigma_{rr}^2 - \sigma_{rr}\sigma_{\theta\theta} - \sigma_{\theta\theta}^2}. \tag{2.40}$$

In accordance with this equation, the admissible pressure before the plastification of the cylinder is $p_{elast} = 0.103$ GPa. Beyond this internal pressure, the cylinder will plastify from the center to the outside and will collapse when the plastic front reaches the outer face.

2.2.6.1 Results

Figure 2.6 shows the Von Mises stress and the plastic strain repartition on the cylinder, for the particular internal pressure of $p_i = 0.14$ GPa. We can see that the plastic yielding started progressively at the inner surface with the increase of the internal pressure and stabilized. Figure 2.7 gives the Von Mises criteria evolution as a function of the radius. It also gives the limit between elastic and platic area at the radius of $r_p = 117$ mm.

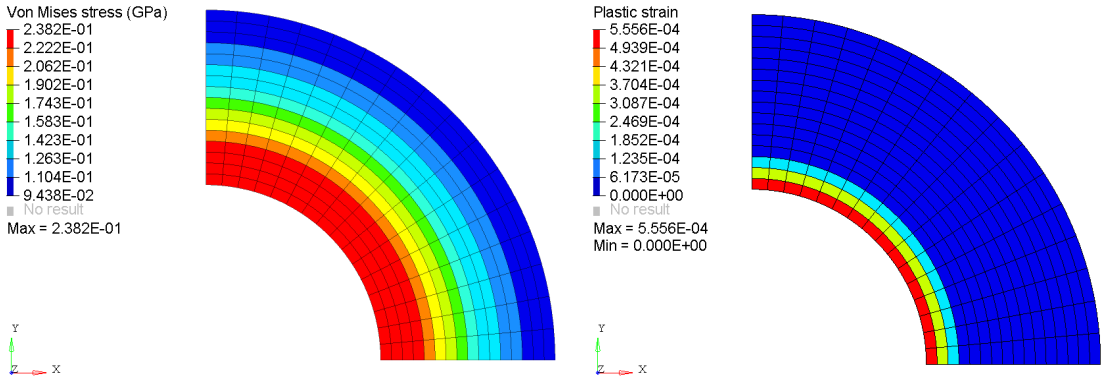


Fig. 2.6: a) Von mises stress, b) Plastic strain after an application of an internal pressure of $p_i = 0.14$ GPa.

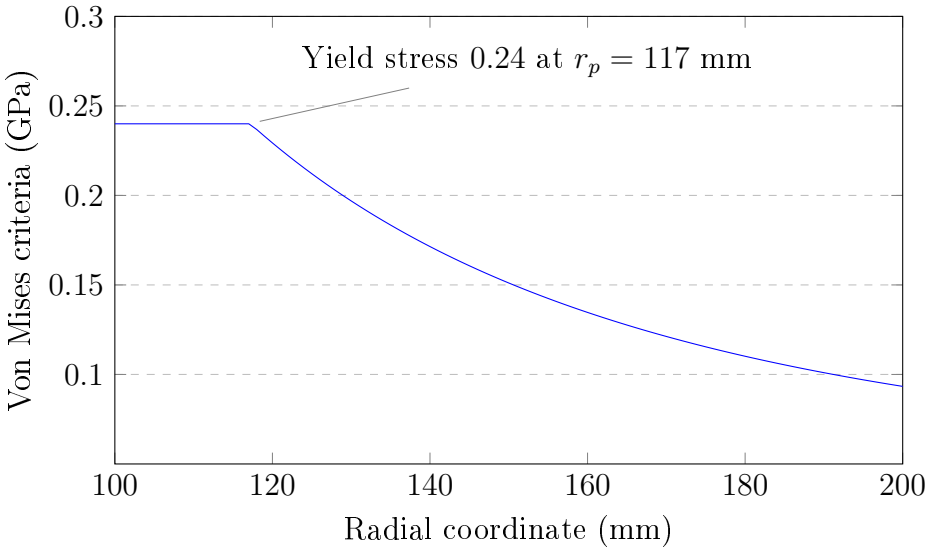


Fig. 2.7: Von Mises criteria evolution depending on the radius of the cylinder.

2.3 Critical time increment for 3D isogeometric analysis

As classical finite element, the implementation of IGA in an explicit solver requires to compute the critical time increment associated to these elements.

2.3.1 Critical time increment estimation methods

Reducing the calculation time of the simulations carried out in the industrial codes is a particularly important point in terms of performance. This computational time is directly related to the time increment estimation at each time step and its maximization is fundamental, while ensuring the stability of the central difference scheme. The good behavior of IGA in the high frequency regime that has been observed by Cottrell et al. [28], Cottrell et al. [27], allows to obtain higher critical time increments than FEA as observed by [14].

For FEA, several estimates are available in the litterature and have been used for IGA such as the nodal time increment [43, 11, 54, 10, 49], the characteristic length method [48, 49], or the power iteration method [105, 11, 14, 15, 16]. These estimates are also available for IGA in Radioss and will be reviewed in the next sections.

2.3.1.1 The characteristic length

In a heuristic way for classical finite elements, the characteristic length L_c, see [48, 49], is defined as follows:

$$L_c = \frac{V_e}{max(S_e)}, \tag{2.41}$$

with V_e the element volume and $max(S_e)$ the maximum element face area.

The specificity of the isogeometric elements is that they can be very distorted. Using this formula as is would give bad time increment values. Furthermore, the difficulty lies in the fact that the computation of the volume or the area with its control points has to be done in the parametric space, then to be switched to the physical space.

One first estimate is to use the minimum volume value associated to the integration points, and multiply it by the number of these points in the element in order to have an equivalent elementary volume, see Figure 2.8. Depending on the strain state of this element, this volume may be smaller or equal to the element volume. The area associated to this equivalent volume will be the maximum median area of the element. It is computed setting as follows:

$$S_{\xi_{median}} = \sum_{j=1}^{m}\sum_{k=1}^{o} \frac{dM_{j,q}(\eta_j)}{d\eta}\frac{dR_{k,r}(\zeta_k)}{d\zeta}\frac{d\eta d\zeta}{dydz} w_{\xi_{median},\eta_j,\zeta_k}, \tag{2.42}$$

for the case of median area in the ξ direction, with m and o the number of Gauss points in the η and ζ direction. The principle is similar for the two other median

areas in the other directions. The subvolumes computation is done as follows, with n the number of Gauss points in the ξ direction:

$$V_{Gp} = \frac{dN_{i,p}(\xi_{Gp})}{d\xi} \frac{dM_{j,q}(\eta_{Gp})}{d\eta} \frac{dR_{k,r}(\zeta_{Gp})}{d\zeta} \frac{d\xi d\eta d\zeta}{dxdydz} w_{Gp}. \tag{2.43}$$

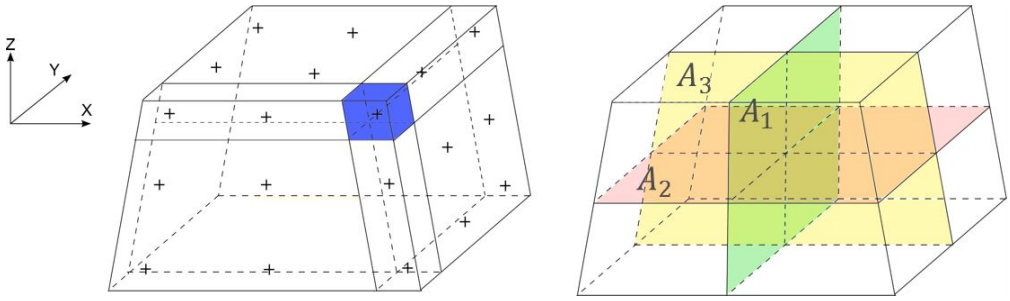


Fig. 2.8: The minimum subvolumes of integration points and median areas in a quadratic B-Spline or NURBS element.

This method brings good results for elements that have a limited number of integration points per direction, i.e. for linear or quadratic basis functions. Beyond this, the minimum volume that is taken into account may become very small because of the geometric distribution and weight of these integration points, and the time increment that is computed would be over-conservative.

A second way to compute the time increment is given by choosing the minimum of the subvolumes constituted on a band of integration points, and by multiplying it by the number of integration points in the perpendicular plane. In this way, the equivalent elementary volume is greater than the first one, while remaining lower than the actual volume. The area which divides this volume must be in this case the maximum area perpendicular to the subvolume band, see Figure 2.9. The following equations give the surface for the superior ξ face and the corresponding subvolume:

$$S_{\xi_-} = \sum_{j=1}^{m} \sum_{k=1}^{o} \frac{dM_{j,q}(\eta_j)}{d\eta} \frac{dR_{k,r}(\zeta_k)}{d\zeta} \frac{d\eta d\zeta}{dydz} w_{\xi_-,\eta_j,\zeta_k}, \tag{2.44}$$

$$V_{Gp}(\xi, \widehat{\eta}, \widehat{\zeta}) = \sum_{i=1}^{n} \frac{dN_{i,p}(\xi_{Gp})}{d\xi} \frac{dM_{j,q}(\widehat{\eta})}{d\eta} \frac{dR_{k,r}(\widehat{\zeta})}{d\zeta} \frac{d\xi d\eta d\zeta}{dxdydz} w_{\xi,\widehat{\eta},\widehat{\zeta}}. \tag{2.45}$$

We have observed that for interior element with maximum continuity, the characteristic length method is very conservative. Indeed a scaling factor of $\sqrt{\frac{4}{3}}$ can for example be used with C^1 quadratic basis functions. However, we have also observed that the critical time increment can be lower than the estimate for elements with reduced continuity. This is particularly the case for the elements

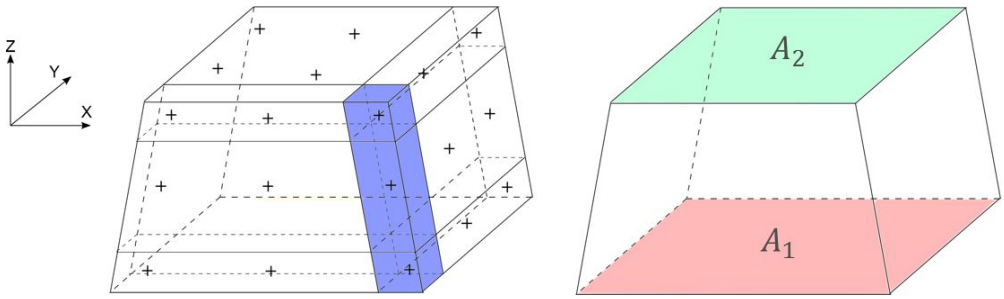


Fig. 2.9: A band of subvolumes of integration points and the two corresponding areas in a quadratic B-Spline or NURBS element.

near the patch boundaries or near areas of reduced continuity.

The proposed method has been implemented in Radioss with a safety factor of $\frac{2}{3}$ that guarantees in most cases a safe estimation even for boundary elements.

2.3.1.2 Nodal stiffness time increment

A non-heuristic approach to estimate the critical time increment is to solve Equation (2.9), i.e. to compute the maximum eigenvalue ω_{max} of the system $\underline{\underline{M}}^{-1}\underline{\underline{K}}$ using the stiffness matrix and the mass matrix, see [43, 11, 54, 10, 49]. Lumped mass matrix $\underline{\underline{M}}$ and stiffness matrix $\underline{\underline{K}}$ have to be assembled. The lumped mass matrix $\underline{\underline{M}}$ is already computed, but the stiffness matrix $\underline{\underline{K}}$ is sparse and not computed in an explicit code. Another lumped matrix $\underline{\underline{\tilde{K}}}$ must be assembled.

One way to express the diagonal terms of the stiffness matrix is to start from its definition which is $\underline{\underline{B^t D B}}$. The compression wave modulus M, see Equation (2.46), can be given as an upper bound of the matrix $\underline{\underline{D}}$.

$$M = K + \frac{4}{3}G = \rho.c_0^2, \tag{2.46}$$

with K the Bulk modulus and G the shear modulus. This sum, equivalent to $\rho.c_0^2$, and the product $\underline{\underline{B}}^t\underline{\underline{B}}$, gives an upper bound of the diagonal term of the stiffness matrix:

$$K_{ii}^e = \rho.c_0^2 \sum_{Gp} Vol_{Gp} \left(|\frac{\partial N_i}{\partial x}| \sum_{j=1}^{ncp} |\frac{\partial N_j}{\partial x}| + |\frac{\partial N_i}{\partial y}| \sum_{j=1}^{ncp} |\frac{\partial N_j}{\partial y}| + |\frac{\partial N_i}{\partial z}| \sum_{j=1}^{ncp} |\frac{\partial N_j}{\partial z}| \right). \tag{2.47}$$

By definition, the lumped stiffness matrix gives a maximum eigenvalue which is larger than the one obtained with the true stiffness matrix:

$$\omega_{max}^{M^{-1}K} \leq \omega_{max}^{M^{-1}\tilde{K}}. \tag{2.48}$$

The eigenvalue of $M^{-1}\tilde{K}$ is then computed as:

$$\omega_{max} = \sqrt{max\left(\frac{\tilde{K_{ii}}}{M_{ii}}\right)}. \tag{2.49}$$

Then, the associated time increment is:

$$\Delta t = \frac{2}{w_c} = \frac{2}{\sqrt{max\left(\frac{K^e_{ii}}{M^e_{ii}}\right)}}. \tag{2.50}$$

Considering the partition of unity, each diagonal term of the stiffness matrix is equal to the sum of the absolute values of the non diagonal terms. It follows another definition of the diagonal values of the lumped stiffness matrix, depending on the non-lumped stiffness matrix:

$$K^e_{ii} = \rho c_0^2 \sum_{Gp} Vol_{Gp}\left((\frac{\partial N_i}{\partial x})^2 + (\frac{\partial N_i}{\partial y})^2 + (\frac{\partial N_i}{\partial z})^2\right) = \sum_{j\neq i} |K^e_{ij}|, \tag{2.51}$$

$$\tilde{K^e_{ii}} = 2K^e_{ii}. \tag{2.52}$$

The eigenvalue definition of $M^{-1}\tilde{K}$ coming from Equation (2.49) becomes:

$$\omega_{max} = \sqrt{2max(\frac{K_{ii}}{M_{ii}})}, \tag{2.53}$$

and

$$\Delta t = \frac{2}{\sqrt{2\frac{K_{ii}}{M_{ii}}}} = \sqrt{2min(\frac{M_{ii}}{K_{ii}})}. \tag{2.54}$$

The same safety factor of $\frac{2}{3}$ is applied to the nodal time increment to be sure that stabitily is preserved.

2.3.1.3 The power iteration method

A third way to estimate an isogeometric time increment is to use the power iteration method. This method was introduced by Wilkinson [105] and used for the first time by Benson [11] for FEA, and later for IGA [14, 15, 16]. This method approximates the largest eigenvalue of a given system taking into account all the boundary conditions, including contact phenomena. To be able to apply this method, $\underline{\underline{M}}^{-1}\underline{\underline{K}}$ has to be symetric and positive-definite. In summary, this matrix must be diagonalizable. In the case of Radioss, the mass matrix is diagonal and meets these conditions. Considering the matrix $\underline{\underline{M}}^{-1}\underline{\underline{K}}$ of dimension $n \times n$, its eigenvalues are supposed to be all different. Thus, $\underline{\underline{M}}^{-1}\underline{\underline{K}}$ has n eigenvalues $\lambda_1, \ldots, \lambda_n$ such as:

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|. \tag{2.55}$$

Defining $\underline{u_1}, \ldots, \underline{u_n}$ the associated eigenvectors. $\underline{x_0}$ is a vector $\mathbb{R}^n$ with magnitude 1 which is decomposed as follows:

$$\underline{x_0} = \sum_{i=1}^{n} x_i.\underline{u_i}. \tag{2.56}$$

The series of $x_{k+1} = (\underline{\underline{M}}^{-1}\underline{\underline{K}}).x_k$ is computed by:

$$\underline{x_k} = (\underline{\underline{M}}^{-1}\underline{\underline{K}})^k.\underline{x_0}, \tag{2.57}$$

$$\underline{x_k} = \sum_{i=1}^{n} \lambda_1^k.(\underline{\underline{M}}^{-1}\underline{\underline{K}})_i.\underline{u_i}. \tag{2.58}$$

Factoring λ_n^k we obtain:

$$\underline{x_k} = \lambda_n^k.(\sum_{i=1}^{n} (\frac{\lambda_i^k}{\lambda_n^k}).(\underline{\underline{M}}^{-1}\underline{\underline{K}})_i.\underline{u_i}). \tag{2.59}$$

Since $\frac{\lambda_i^k}{\lambda_n^k}$ tends to 0, the predominant term in the previous equation is $\lambda_n^k.(\underline{\underline{M}}^{-1}\underline{\underline{K}})_n.\underline{u_n}$. After a few iterations we estimate the highest eigenvalue:

$$|\lambda_n| \backsimeq \frac{\|\underline{x_{k+1}}\|}{\|\underline{x_k}\|}, \tag{2.60}$$

and the associated eigenvector:

$$\underline{u_n} \backsimeq \underline{x_k}. \tag{2.61}$$

If we end up in the case where $|\lambda_n|$ is too high, we can normalize the serie, following:

$$\underline{x_k} = \frac{\underline{x_k}}{\|\underline{x_k}\|}. \tag{2.62}$$

The time increment associated to the highest eigenvalue $|\lambda_n|$ is:

$$\Delta t = \frac{2}{|\lambda_n|}. \tag{2.63}$$

As such, the power iteration method is computationally too expensive to be used at each time step. A more efficient way in terms of computational cost compared to time increment gain is to use a second estimate of the critical time increment, such as nodal stiffness or elemental stiffness, and to establish a ratio $r = \Delta t_{max}^{PIT}/\Delta t_{max}^{element}$. Benson [11] proved that this ratio changes relatively slowly compared to the time increment value for most nonlinear cases. Only one or two

iterations are sufficient to update the time increment with the power iteration and we will check that this ratio r does not change too much. The calculation of this ratio is much less expensive than the complete calculation of the power iteration. As long as this ratio remains approximately constant, we can consider that the estimated time increment $\Delta t_{max} = r \times \Delta t_{max}^{element}$ is valid. In case of variation beyond a limit error, new iterations must be performed to readjust the time increment and this ratio. According to our observations on nonlinear simulations, it takes about twenty iterations at the beginning of a computation to have a good value, then at each update, only one or two iterations are enough to readjust the value.

In practice, the implementation of the power iteration in Radioss safely limits the amplification factor of the time increment to 1.2 times per time step. The estimate of the maximum natural frequency is automatically deactivated if the ratio r obtained does not reach a given minimal value. This saves computing costs for problems with too strong nonlinearities and for which the power iteration is no longer interesting. In this way, this estimate of the critical time increment can be constantly used during the simulation. This method combining two estimates of time increments makes it possible to minimize the costs of the power iteration while benefiting from a gain that remains important. To ensure the robustness of the estimation in cases which have strong non-linearities and do not cause the system to diverge, a scaling coefficient of 0.9 is still applied to the resulting time increment.

2.3.1.4 The boundary influence on the time increment

Adam et al. [1] studied the fact that boundary elements penalize the critical time increment due to the reduced length of the basis function support. As only open knot vector are used, the boundary functions have a C^0 continuity, which is reduced compared to interior functions. The element time increment is directly impacted by the boundary elements size, and the nodal time increment is impacted by the accumulation of control point near the boundary. As such the external elements drive the time increment, no matter the estimation method. One solution could be to expand the size of these elements. Another solution is to use mass scaling techniques [81, 82, 25, 83, 49]. These methods modify the material properties such as the density of the smallest elements, generally the external ones, to involve a mass improvement and directly improve the time increment. However, this modification can change the physics of the problem and can reduce the quality of the results.

2.3.2 Critical time increment comparison: Square Taylor bar impact

A comparison of the time increment estimates was done in 1D for uniform and non uniform mesh by Adam et al. [1]. To compare the accuracy of these estimates on a

three-dimensional problem, a Square Taylor bar impact simulation is introduced.

2.3.2.1 Model definition and boundary conditions

This simulation consists in a square rod impacted on a rigid wall with an initial velocity of 227 m.s^{-1}. The rod is 32.4 mm long and has a side of 3.2 mm. This simulation can be refered to experimentations and numerical studies done for cylindrical rod cases by Rakvåg et al. [88, 87]. Several materials and impact velocities are given in theses experimentations and allow to see an extended range of behavior, see Figure 2.10(b). Experimentations done for cylindrical rod show that the high velocity impact causes severe deformation at the rod's extremity. It will cause mesh distortion in the corresponding simulations for elements near the lower end of the rod, dramatically reducing the stable time increment during the solution. We have used the square version of the Taylor bar for comparison with the reference example available in the Radioss database.

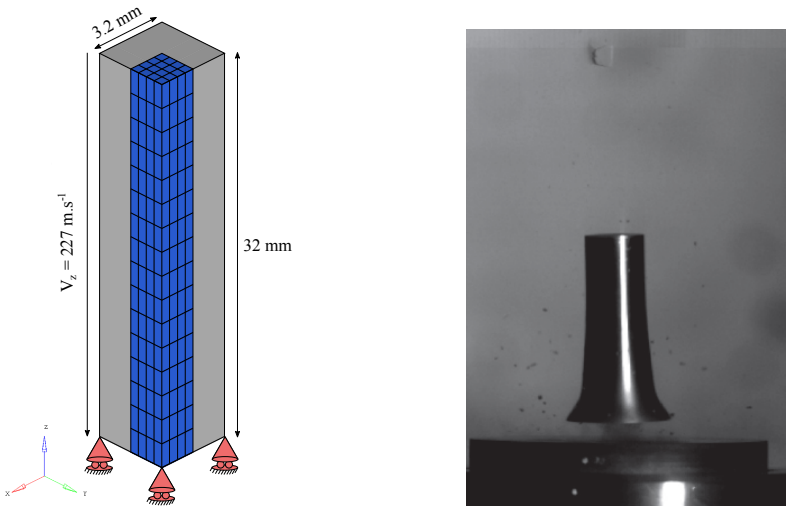


Fig. 2.10: a) Boundary conditions and B-Splines quadratic C^1 mesh. b) High-speed video image from Taylor bar impact tests with nominal impact velocity of 300 m.s^{-1}. The image shows typical impact for an unhardened cylindrical projectile with mushroom deformation, from [88].

In this case, an isotropic elasto-plastic material with a Johnson-Cook plasticity model is used to represent the copper behavior. It is a Von Mises elastic, perfectly plastic material. The Young's modulus is set to 110 000 MPa, the yield stress is defined as 314 MPa, the density is 8 970 kg.m^{-3} and the Poisson ratio is 0.3. The boundary conditions and dimensions are shown in Figure 2.10(a). The model used

here is a C^1 quadratic B-Spline patch which represents one quadrant of the rod, due to the inherent symmetry. The B-Spline patch is composed of 256 elements. Corresponding knot vectors are given in Table 2.6. The appropriate boundary conditions prescribed on each of the two symmetry planes for the problem are defined. On the bottom face of the rob, the vertical component of the kinematic fields is fixed. The other components are free. An initial vertical velocity of 227 m.s^{-1} is imposed on all the control points.

Direction	Order	Knot vector
ξ	p=2	$\Xi = \left[0,0,0,\frac{1}{4},\frac{2}{4},\frac{3}{4},1,1,1\right]$
η	q=2	$H = \left[0,0,0,\frac{1}{4},\frac{2}{4},\frac{3}{4},1,1,1\right]$
ζ	r=2	$Z = \left[0,0,0,\frac{1}{16},\frac{2}{16},\ldots,\frac{14}{16},\frac{15}{16},1,1,1\right]$

Table 2.6: Knot vectors for the square Taylor bar impact mesh.

Kamoulakos [63] gives the total equivalent plastic strain expression at the center of the rod for the cylindrical rod case:

$$\varepsilon_p = \frac{C_E^2 - C_p^2}{C_E^2 C_p}(V - \frac{Y}{\rho C_E}) = 2.02, \tag{2.64}$$

with C_E and C_p respectively the elastic and plastic waves speeds and V the initial velocity of the rod. This value will be considered as a reference value in the square rod case.

2.3.2.2 Results

Simulation result is shown in Figure 4.2 and reproduce the mushrooming behavior observed in [88]. From this figure it is clear that extremely high plastic strains develop at the crushed extremity of the rod, close to the center of the bar, and result in severe local mesh distortion.

Power iteration estimates of critical time increment and related variables are given in Figure 2.12. Figure 2.12(a) gives the power interation time increment estimated during the simulation and Figure 2.12(b) the ratio between power iteration time increment estimate and nodal stiffness one. Figure 2.12(c) and Figure 2.12(d) give the number of iterations needed for the power iteration algorithm to converge to the maximum eigenvalue of the model.

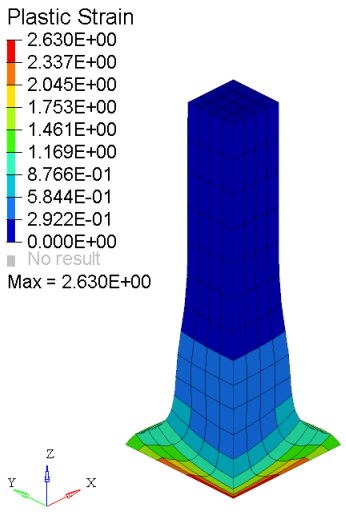


Fig. 2.11: Plastic strain for the C^1 quadratic NURBS mesh.

We see on these figures that the maximum eigenvalue of the Taylor bar considerably increases when the bar is crushed, see Figure 2.12(d). The number of iterations needed to converge also increases, see Figure 2.12(c), but the power iteration still gives a higher time increment estimate compared to the nodal stiffness one, see Figure 2.12(b).

Another model composed of 20 nodes quadratic FE (S20 in Radioss element library) is introduced. Element density is identical as the B-Splines model, to compare the computational time and the quality of the given solution. It can be noticed that for the same density of elements, the B-Spline model is composed of 648 control points whereas the FE mesh is composed of 1505 nodes, due to the C^1 continuity of the B-Spline functions. All the boundary conditions and kinematic fields are identical to the previous ones. Figure 2.13 shows plastic strain results for both models, and Figure 2.14 gives the evolution of time increment during the run for B-Spline model with the three estimates and for FE for comparison. Safety factors are used for each estimate, in accordance to the classical usage in Radioss, i.e, 0.67 for the nodal and element time increment, and 0.9 for the power iteration. When looking at Figure 2.14, it is clear that nodal time increment and element time increment are overconservative compared to the power iteration one which is continually increasing until the crushing of the Taylor Bar. We see steps for the time increment calculated with the power iteration that come from the limit imposed on the amplification factor set at 1.2 per time step. The time increment given by all the estimation methods is then reduced dramatically over the course of this analysis as a result of the large changes in element aspect ratio. The power iteration method allows to have an optimal time increment all allong the simulation and reduces the total computational time. No matter the estimate used, the

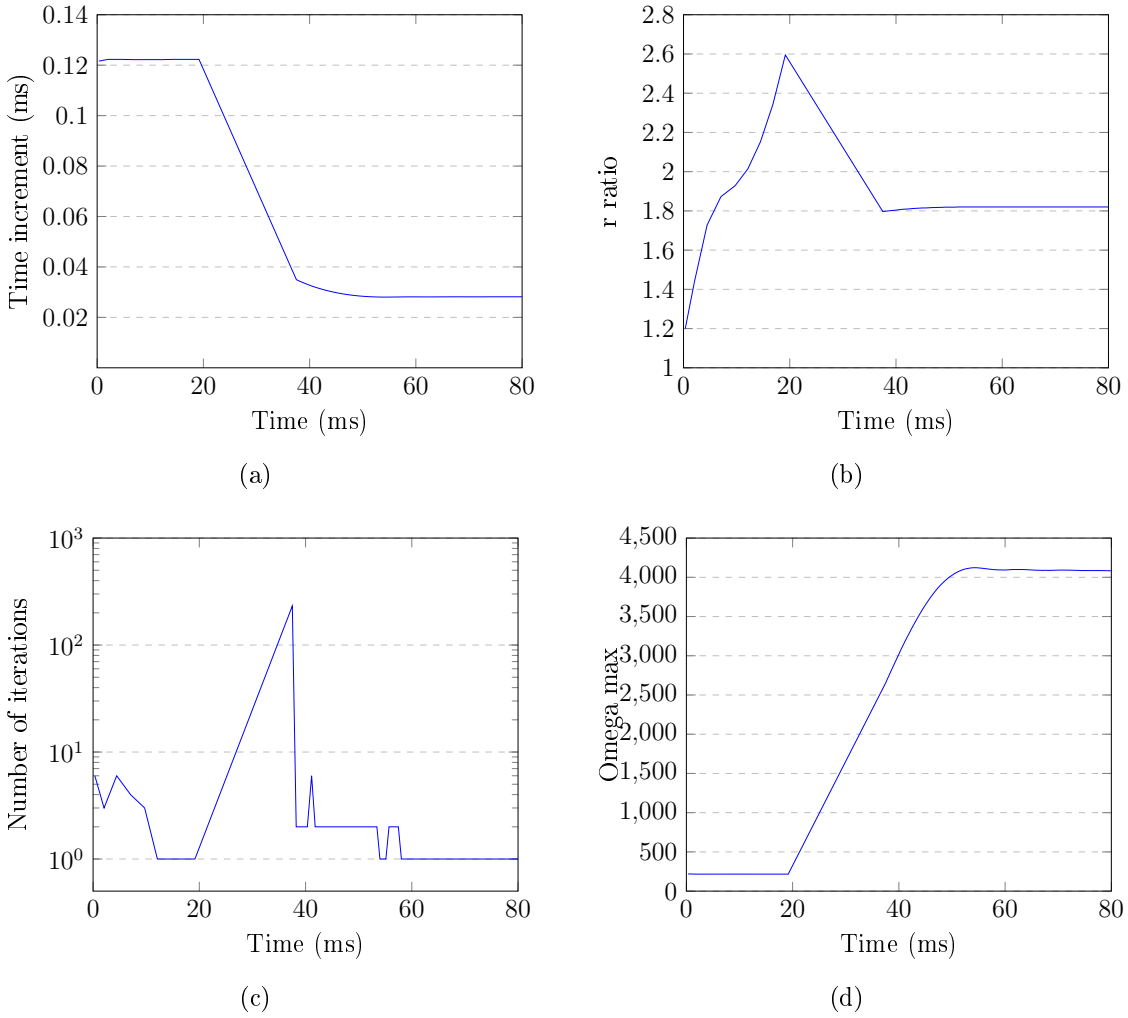


Fig. 2.12: Power iteration estimates of critical time increment and related variables.

B-Spline model has a time increment quite higher than the FE one. The number of element is the same between the B-Spline model and the FE one, however the C^1 continuity inside the patch allows us to have fewer control points but also allows to obtain a better time increment.

Table 4.1 groups information about all meshes in terms of number of elements and number of nodes/control points. It also gives minimal time increment during the simulation, computational time and total number of time steps in the case where the scaling factor applied to the estimated time increment is set to its higher value, i.e., setting at the limit of divergence. In this case the time increment will not be constant. The higher value that can be reached will allows to compare the efficiency of the model and the basis functions without taking into account the overconstraining safety factor. For comparison purposes, the results obtained with

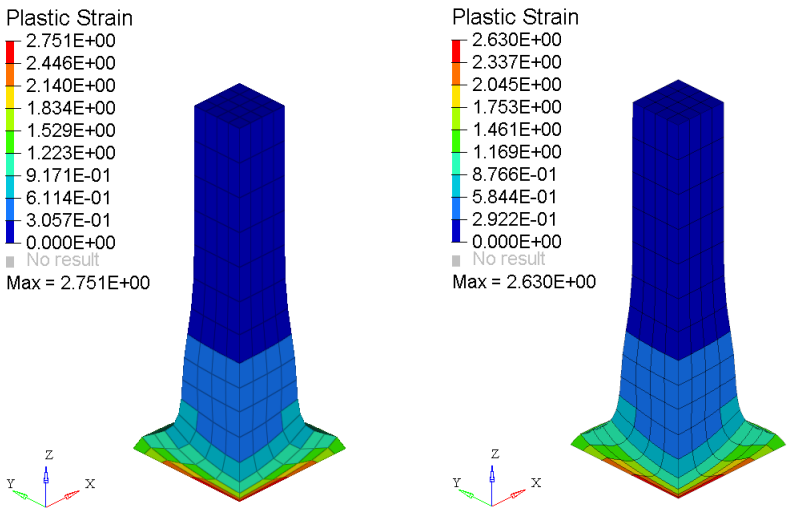


Fig. 2.13: Plastic strain: a) C^1 quadratic B-Spline mesh, b) 20-nodes Lagrange quadratic mesh.

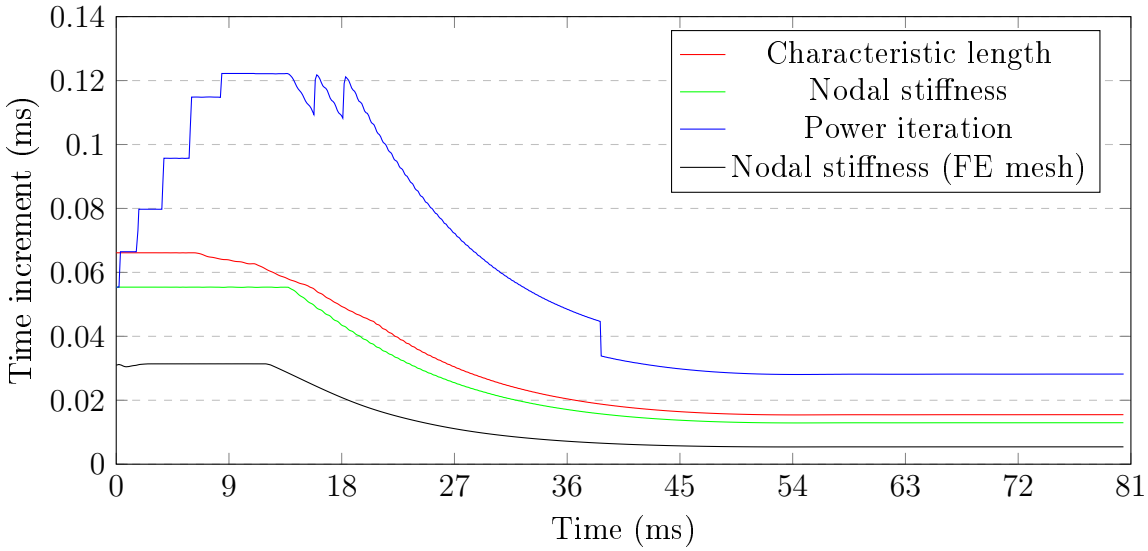


Fig. 2.14: Evolution of time increment for each estimate method for the B-Spline model and nodal stiffness estimate for the FE model.

the finite element model are considered as a reference, values for other models are normalized with respect to these ones. This table shows that B-Spline based model can be at least 40% faster, mainly due to a much higher time increment, about 6 times higher, which greatly reduces the total number of time steps. The quality of

the obtained solution is also better for the B-Spline model, with a maximum plastic strain closer to the reference value.

	S20 Lagrange	B-Spline Quad C^1
Number of elements	256	256
Number of nodes	1505	648
Relative time increment	1	6.25
Relative number of time steps	1	0.28
Relative total CPU time	1	0.56
Maximum Plastic Strain	2.751	2.630

Table 2.7: Comparison table for both meshes in the case of the security scaling factor applied to the time increment is set to its maximum value.

2.4 Isogeometric contact

Since its introduction, IGA showed the capacity to better represent complex geometries and to increase the accuracy and robustness compared to standard finite element method. One of the areas of interest is contact mechanics, in which the higher continuity of basis functions and smooth geometry description have potential advantages in the description of interacting surfaces undergoing large displacements and large sliding. Several contact formulations within the IGA framework were introduced in this framework and were presented in the first chapter. Their complexity and their easy-to-implement aspect in industrial codes is to be taken into account when considering them, especially in Radioss which is a rather complex existing code. In the context of this study, an existing node to segment contact interface was chosen to integrate the B-Spline models. This choice allows to benefit from the efficiency of the existing algorithms, like sorting and contact candidates selection, by minimizing the programming effort required. The so called "Type 7 contact interface", see [106], has been chosen. The contact formulation in the Radioss Type 7 interface is based on the node to segment formulation and uses a master/slave treatment with nonlinear penalty. Master segments are defined depending on the type of element they lie on. If it is a 3-node or a 4-node shell, the segment is the surface of the element. If it is a solid element, the segment is defined as the outer face. This formulation is similar to the single-pass algorithm introduced by Matzen et al. [75].

2.4.1 Isogeometric external surface fitting

In the case of B-Spline or NURBS patches, the fitting of the outer surface follows the bilinear quadrilateral collocation scheme introduced in [13, 14]. The fitting is done at the element level introducing a set of so called ghost points which define bilinear segments. Each isogeometric element is fitted by 16 ghost points for one element outer face, i.e. 4 ghosts points by direction, according to a constant step into the knot interval of the element which depends on the normal direction of the outer element face, see Figure 2.15.

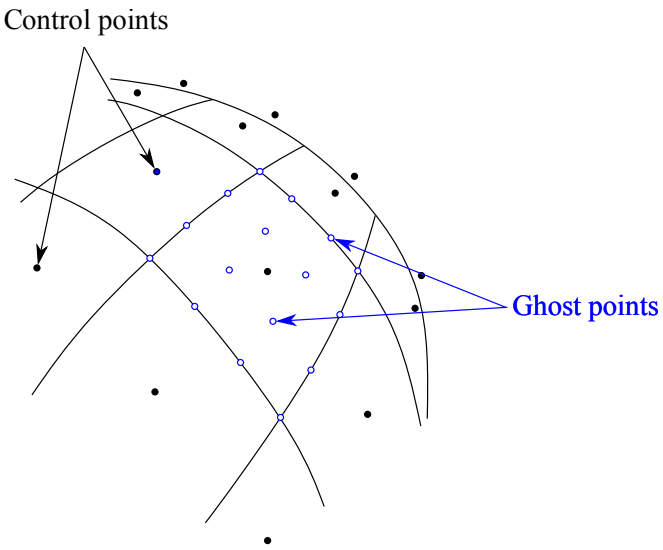


Fig. 2.15: Fitting of the surface of a quadratic B-Spline geometry. The Spline surface and the control points are drawn in black. The blue points are the ghost points.

The knot corresponding to the parametric normal direction of the element is set to $+1$ or -1, depending on the outer side. The two other knots take the following values, for the i^{th} element, with $j \in [1, ..., 4]$:

$$\begin{aligned} \xi_j^{ghost} &= \xi i + \frac{(j-1)step_\xi}{3}, \\ \eta_j^{ghost} &= \eta i + \frac{(j-1)step_\eta}{3}, \\ \zeta_j^{ghost} &= \zeta i + \frac{(j-1)step_\zeta}{3}, \end{aligned} \tag{2.65}$$

with

$$\begin{aligned} step_\xi &= \frac{\xi_{i+1}+\xi_i}{3}, \\ step_\eta &= \frac{\eta_{i+1}+\eta_i}{3}, \\ step_\zeta &= \frac{\zeta_{i+1}+\zeta_i}{3}. \end{aligned} \tag{2.66}$$

Physical coordinates and velocities are then computed for each ghost point:

$$\underline{x}^{ghost} = \sum_{j=1}^{nctrl} N_j(\xi^{ghost}, \eta^{ghost}, \zeta^{ghost})\underline{x}_j, \tag{2.67}$$

$$\underline{v}^{ghost} = \sum_{j=1}^{nctrl} N_j(\xi^{ghost}, \eta^{ghost}, \zeta^{ghost})\underline{v}_j. \tag{2.68}$$

Updating points coordinates. These ghost points lie on the physical surface and are to be moved in accordance to the surface displacements during the the simulation. The update of these coordinates requires a similar computation than for their creation, using the associated stored parametric index.

Slave and master side segment creation. The 16 ghost contact points of one element face are directly used for the slave side. 9 segments are then constructed for each isogeometric element external face from the ghost contact points on the master side. Figure 2.16 shows the red ghost points created for one slave element according to its parametrisation and its control points. On the master side, the blue ghosts points also created according to the element define all the 9 master segments, also plotted in blue.

2.4.1.1 Contact nodes merging

The surface fitting procedure will create some duplicated ghosts points, since some adjacent elements allow the creation of several ghost points located geometrically in the same position, as can be seen in Figure 2.17. In that case, a research and sorting routine is needed for geometric duplicates after the ghost point generation.

This routine performs a sorting according to the three real coordinates of the ghost points which then makes it possible to quickly eliminate the duplicates. Sorting is based on a well-known algorithm called quicksort, see [53]. Unidimensional quicksort is an efficient sorting algorithm that is commonly used in industrial codes. This algorithm is modified so that the sorting is done hierarchically according to the three coordinates of the points in space. The result of this sorting is an elimination of these duplicates and the number of ghost points decreases quite significantly, but the link between a ghost point and some parent elements is lost.

2.4.1.2 Data structures

The ghost points data associated with the isogeometric surfaces are stored in an allocated structure, see Figure 2.18, which contains several tables:

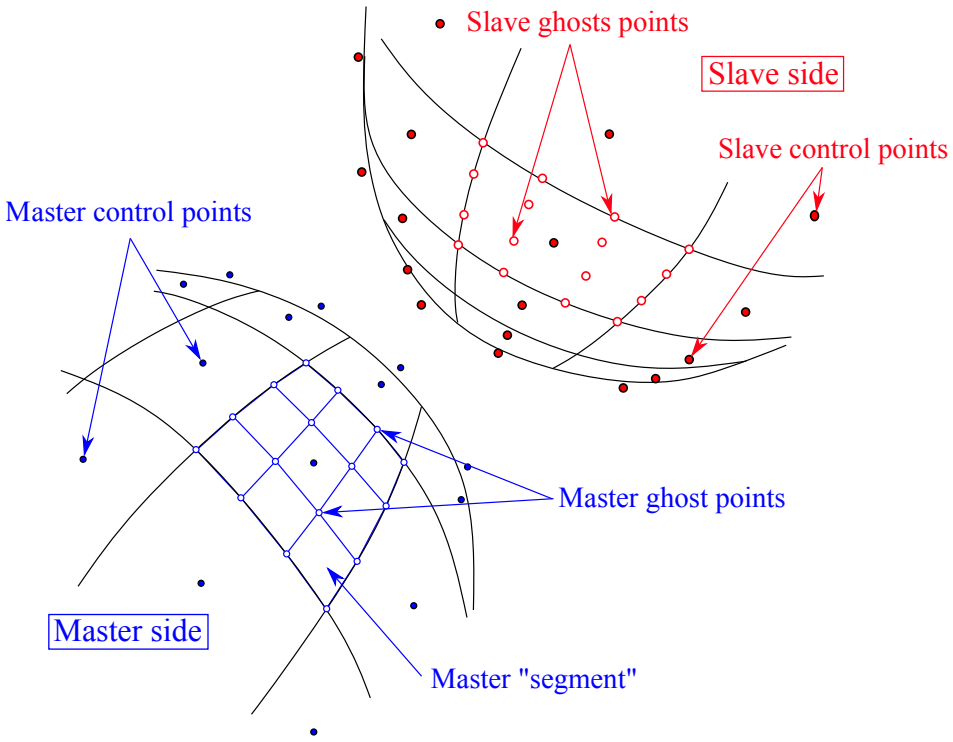


Fig. 2.16: Definition of the slave and the master surface of two quadratic B-Spline geometries. The Spline surface and the control points are drawn in black. The blue side is the master side with the segments built with the ghost points. The red side is the slave side with the associated ghost points.

- The XIVE and VIGE tables contain all three coordinates and three velocities in the physical space of each ghost point;
- The NIGE array contains the number of the parent element of each ghost point. It allows to differentiate points from one element to another, or from one patch to another;
- The RIGE table gives each ghost point position by real values corresponding to its the position with respect to its parent element ($[-1\ 1]; [-1\ 1]; [-1\ 1]$).

2.4.1.3 Contact stiffness distribution

The computation of the contact stiffness is done in a routine which treats each of the interfaces. This procedure makes it possible to define an interface stiffness that can be related to the stiffness of the slave ghost points or to the master ghost segments, depending on the option.

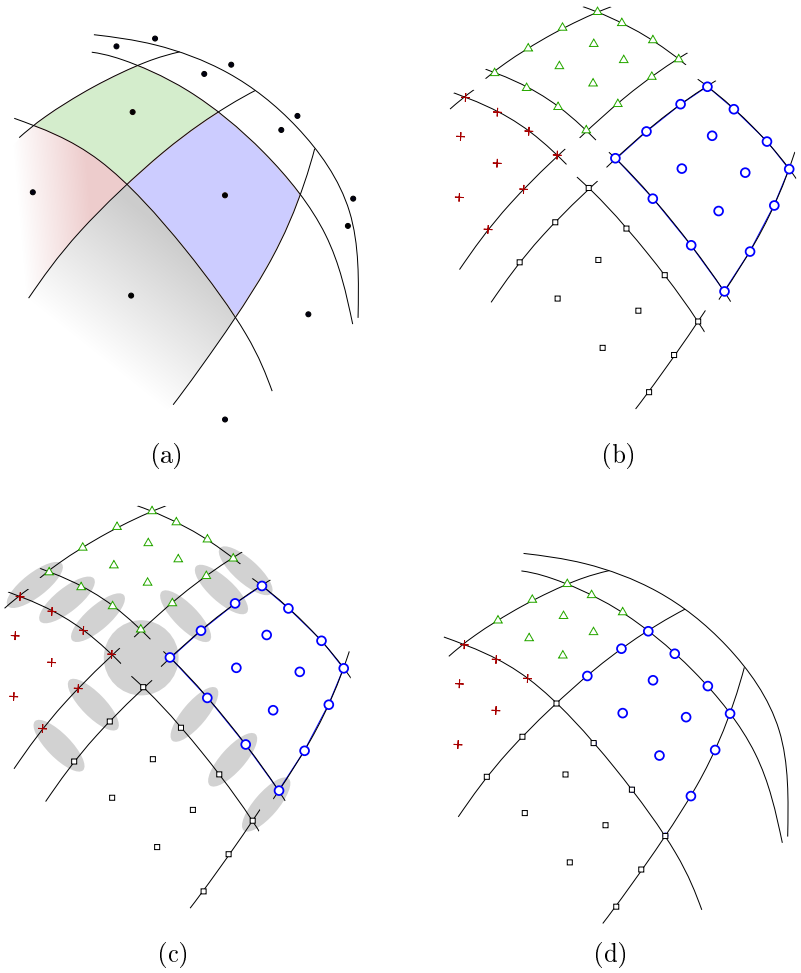


Fig. 2.17: Merging procedure of the ghost contact points created to fit the physical contact surface and eliminate the geometricaly duplicated ones. a) B-Spline contact surface with the corresponding external control points and four elements to be fitted, b) Ghost contact points creation on the physical surface for each of these four elements, c) set of geometricaly corresponding ghost contact points at the edge of elements which have to be merged, d) result of the merging according to the physical coordinates of the ghost contact points.

2.4.1.4 Disadvantages of the non-smooth surface

The disadvantages coming from the non-smooth discretization of B-Spline surfaces are not excluded. The B-Spline surface discretization costs will have to be added to the total computational costs of the interface. However, the quality of this discretization and the number of contact segment and contact ghost nodes should improve the robustness and the overall performance compared to a contact with the same element density in conventional FEA.

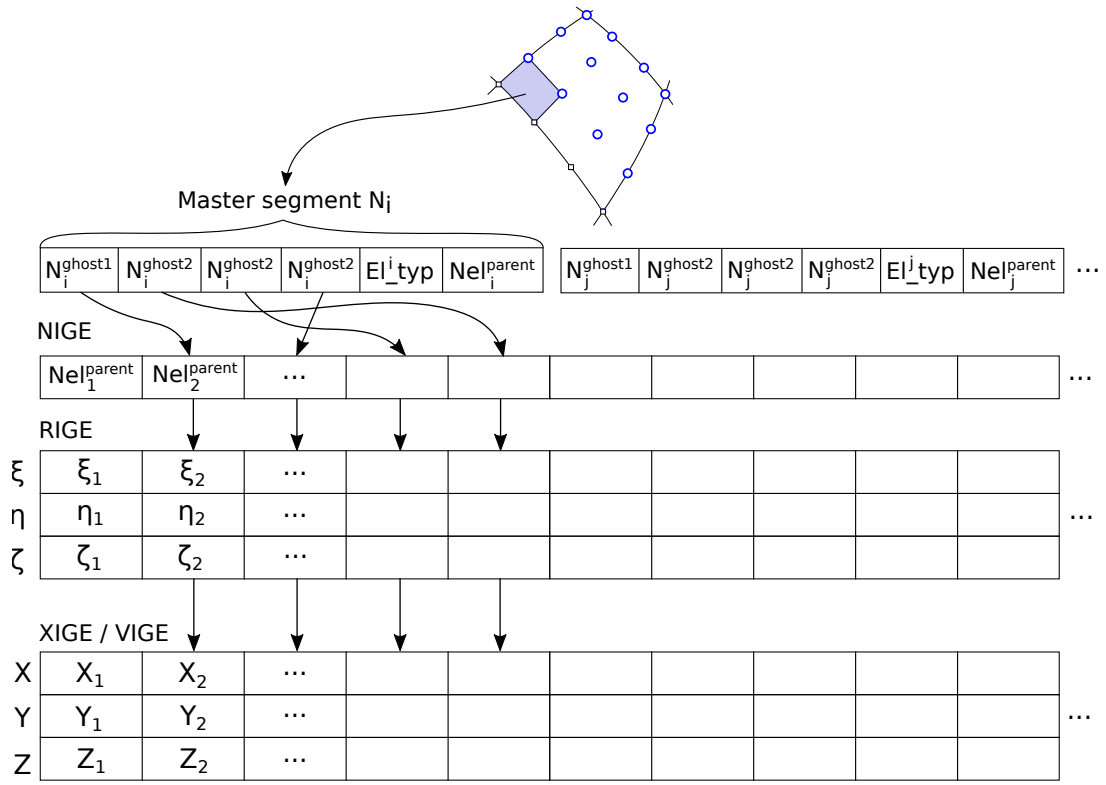


Fig. 2.18: Data structure for ghost contact points definition according to the master or the slave side.

Moreover, the results of the node to surface algorithm, also called single-pass algorithm [75] is strongly dependent on the discretization and on the choice of slave and master surfaces. Some common recommandations specify that it is necessary to have a finer slave discretization than the master one to avoid penetrations.

A double-pass version of the node to surface algorithm [13, 14] can also be defined to avoid these penetrations but it make the system overconstrained. A single-surface algorithm [12] which corresponds to a two-pass node to surface formulation combined with an efficient contact search scheme can be a solution to obtain a non-overconstrained model.

2.4.2 Node to surface contact for isogeometric quadrangular segments

The defined ghost points on the master side will be used to define an isopamaretric space according to the segment:

$$\begin{aligned} H_1 &= (1-s)(1-t)/4, \\ H_2 &= (1+s)(1-t)/4, \\ H_3 &= (1+s)(1+t)/4, \\ H_4 &= (1-s)(1+t)/4. \end{aligned} \tag{2.69}$$

The distance between a ghost slave point and a segment is computed making a projection of the ghost point on the segment and measuring the distance between them. The isoparametric coordinates $(\underline{s}, \underline{t})$ of the projected point are computed using the quadrangular segment's coordinates and a system of two equations, see Equation (2.70).

$$\begin{aligned} \underline{ON_i} &= H_1\underline{ON_1} + H_2\underline{ON_2} + H_3\underline{ON_3} + H_4\underline{ON_4}, \\ \underline{s_i} &= (1-t)\underline{N_1N_2} + (1+t)\underline{N_4N_3}, \\ \underline{t_i} &= (1-s)\underline{N_1N_4} + (1+s)\underline{N_2N_3}. \end{aligned} \tag{2.70}$$

The normal vector $\underline{N_iN_s}$ at the point N_i is then computed with the cross product of the vectors $\underline{s_i}$ and $\underline{t_i}$, see Figure 2.19 and Equation (2.71).

$$\underline{n_i} = \frac{(\underline{s_i} \times \underline{t_i})}{\left\|\underline{s_i} \times \underline{t_i}\right\|}. \tag{2.71}$$

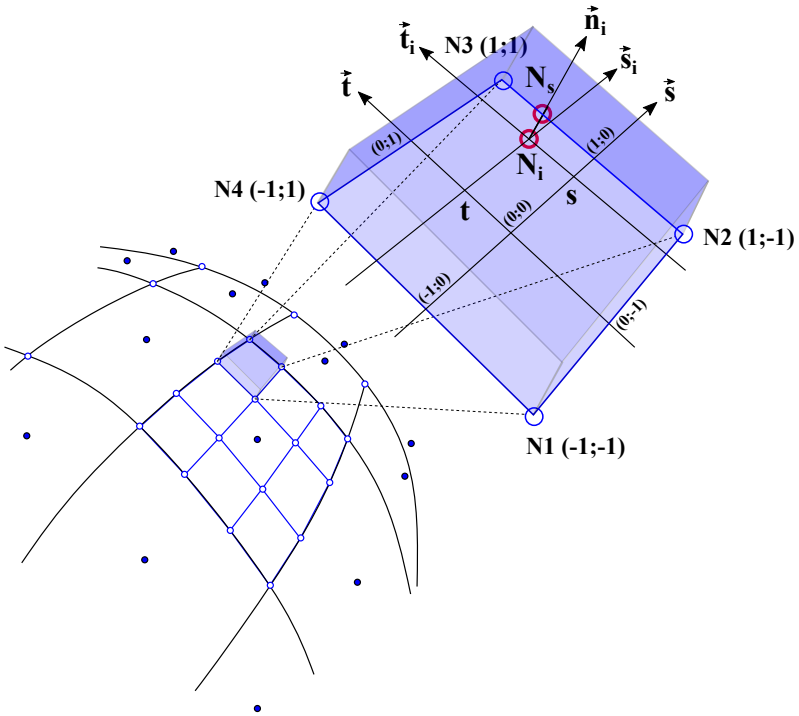


Fig. 2.19: Calculation of the normal vector on the projected point.

The distance from the slave ghost point to the segment is defined as:

$$D = \left\| \underline{N_i N_s} \right\|. \tag{2.72}$$

The contact is activated if the distance between the node and the segment becomes smaller than the gap, i.e. if the node penetrates within the gap, given by $P = max(0, Gap - D)$. The penalty method estimates the normal component of the contact force as a nonlinear penalty force:

$$F = K \times P, \tag{2.73}$$

with the penetration P and the nonlinear stiffness K. The initial value of K is calculated using a combination between stiffness of each slave and master side. After initial penetration, the stiffness is given as a function of the penetration distance and the rate of penetration. The resulting contact force is then applied to the node N_s, see [106].

This computed reaction force and the contact stiffness have to be distributed between the four nodes N_1, N_2, N_3 and N_4 on the master side according to the corresponding shape functions, see Figure 2.20, following:

$$\underline{F}_{i=1,2,3,4} = N(s_i, t_i)\underline{F}, \tag{2.74}$$

where $N(s_i, t_i)$ is the interpolation function for the quadrangular segment. The equation is similar for the stiffness repartition. We have

$$\underline{F_1} + \underline{F_2} + \underline{F_3} + \underline{F_4} = -\underline{F}. \tag{2.75}$$

2.4.2.1 Contact forces repartition at control points

The contact forces and the contact stiffnesses located at the 4 ghost points of the contact segments on the master side, given in Equation (2.74) are brought back to the control points of the element thanks to the NIGE and RIGE table which link the ghost points to the parent element, see Figure 2.21:

$$\underline{F}_i^{cont} = \sum_{i=1}^{nctrl} N_i(\xi^{ghost}, \eta^{ghost}, \zeta^{ghost})\underline{F}, \tag{2.76}$$

$$\underline{K}_i^{cont} = \sum_{i=1}^{nctrl} N_i(\xi^{ghost}, \eta^{ghost}, \zeta^{ghost})\underline{K}. \tag{2.77}$$

On the slave side, the process is the same with the ghost point associated to the master segment. The contact forces and stiffnesses are added to the internal forces and stiffnesses to solve the equilibrium equation.

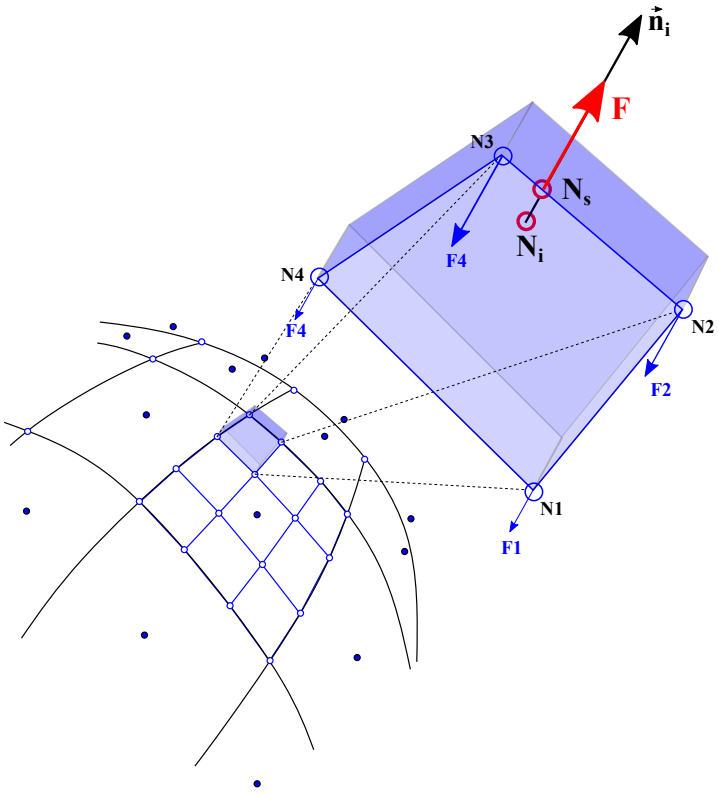


Fig. 2.20: Contact force repartition on ghost points for master side.

2.4.2.2 Self-contact and hybrid contact FE-IGE

The Type 7 interface can be used for self-contact by defining only one surface that will be both described as a master and slave surface. The ghost points of the patch may therefore be candidates for contact with segments that are adjacent to it, with a certain proximity limit. This interface will then operate as two symmetrical interfaces between two different geometries. With the Type 7 contact interface, both IGA and FEA element can be used. It allows to compute hybrid models with contact between IGA and FEA geometries. Several simulations cases will be introduced in Chapter 4, and will illustrate self-contact or hybrid contact.

2.4.2.3 Contact time increment

The contact interface stiffness will be added to the stiffness associated with the element control points, and by definition will impact the critical time increment depending on the intensity of the contact. The more abrupt is the contact, the more important is the penetration and the nonlinear interface stiffness. It will decrease the nodal time increment. The interface uses nonlinear penalty, that is to say that it will benefit from the non-penetration of the slave surface on the master surface provided that both meshes are properly prepared. The element density should be

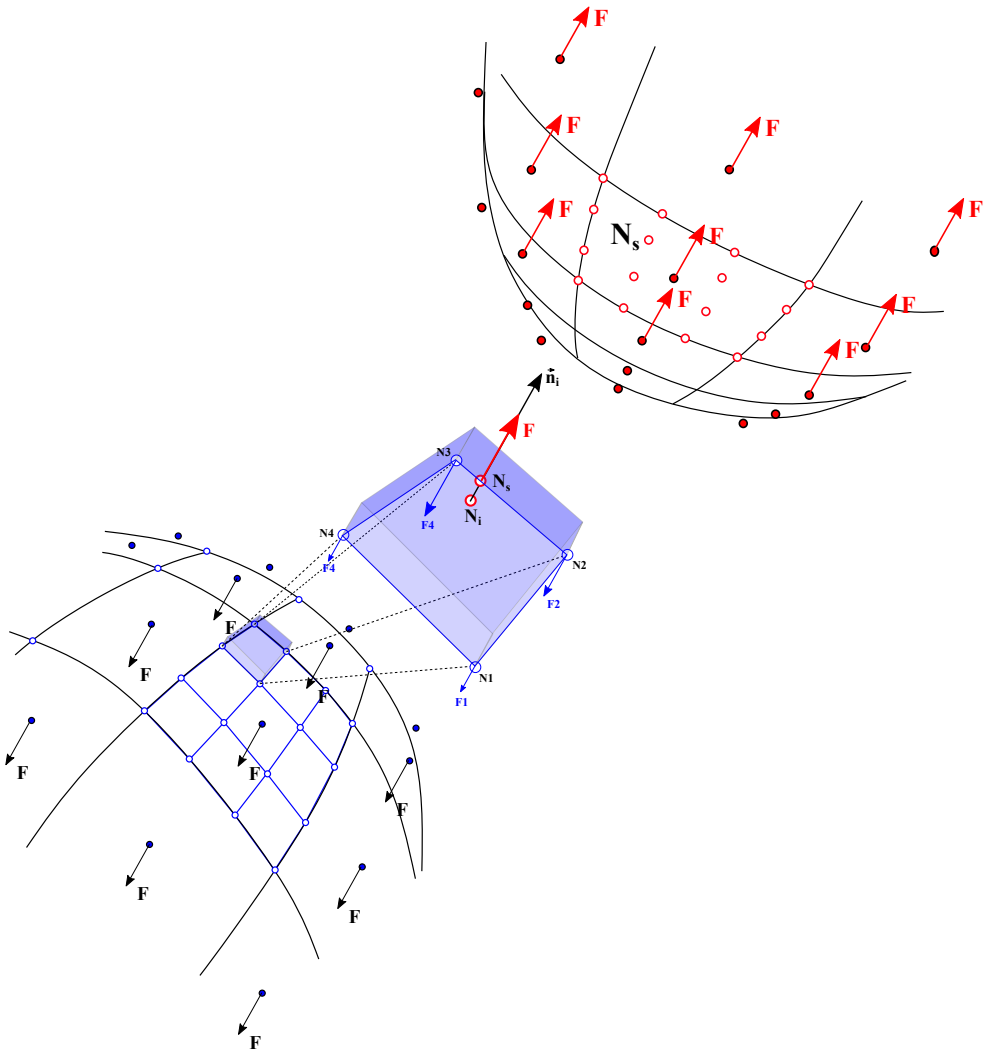


Fig. 2.21: Contact forces repartition at control points for each side.

larger for the master side than for the slave side to limit the penetration values. A kinematic time increment is nevertheless associated with the contact interface, defined as follows:

$$\Delta t^{contact} = \frac{D}{2V}, \tag{2.78}$$

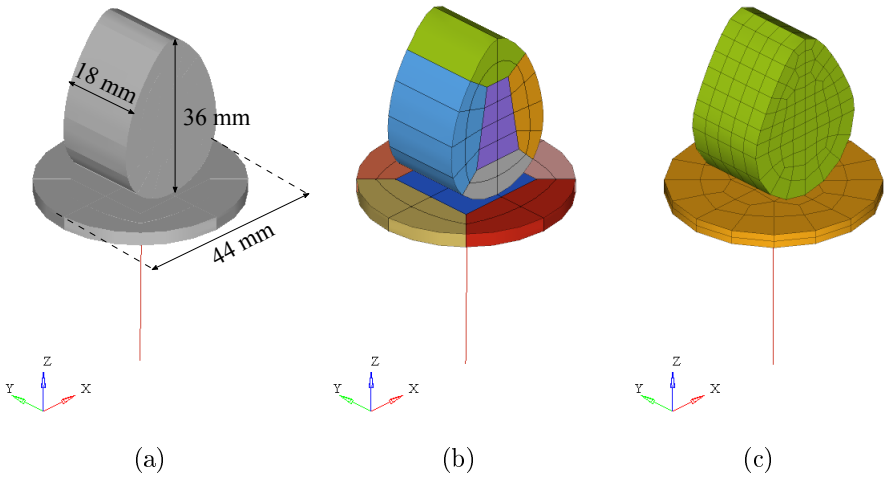
with V the velocity of the slave node and D the distance between the slave node and the master segment. In the case of the contact would be to strong, this kinematic time increment will allow to keep a correct time increment while preventing the slave node to penetrate the master surface.

2.4.3 Isogeometric Contact: Cam-Valve system

In order to test the previous algorithm in the Radioss solver and to illustrate its ability to simulate a dynamic behavior and a kinematic motion, a cam-valve system is introduced.

2.4.3.1 Model definition and boundary conditions

The camshaft of a car takes the rotary motion of the engine and translates it into the linear motion required for operating the intake and exhaust valves. The cam is rotating with an angular velocity of 314 rad.s^{-1}, which interacts and triggers the translation of a valve tied to two springs, attached to the two valves. The superposed springs have varying stiffness, 30 000 N.m^{-1} for the first and 15 000 N.m^{-1} for the second one. The springs control the higher and lower angular frequencies. The valve has a radius of 44 mm, the cam has a height of 36 mm and a width of 18 mm. Boundary conditions and dimensions of this model are given in Figure 2.22.



(a) (b) (c)

Fig. 2.22: Boundary conditions and meshes. a) Cam and valve dimensions, b) C^1 NURBS quadratic mesh and c) 8-nodes Lagrange linear mesh.

The cam and the valve are both meshed with 5 quadratic NURBS patches. The corresponding knot vectors are similar for each patch and are given in Table 2.8. A FE mesh is also defined with 8-nodes bricks (HA8) to compare the contact representation and the obtained results.

Direction	Order	Valve's knot vector	Cam's knot vector
ξ	p=2	$\Xi = \left[0,0,0,\frac{1}{2},1,1,1\right]$	$\Xi = \left[0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1\right]$
η	q=2	$H = \left[0,0,0,\frac{1}{2},1,1,1\right]$	$H = \left[0,0,0,\frac{1}{2},1,1,1\right]$
ζ	r=2	$Z = [0,0,0,1,1,1]$	$Z = [0,0,0,1,1,1]$

Table 2.8: Knot vectors for each patch of the cam and valve models.

The material used for the cam and the valve is steel. For this simulation, an isotropic elastic model is used. Steel has an initial density of 7.8×10^{-3} g.mm^{-1}, a Young modulus of 210 000 MPa, a Poisson ratio of 0.3. The Type 7 interface is used to model contact between the valve's planar surface and the cam's curved one.

2.4.3.2 Results and observations

The kinematics of the simulation is shown on Figure 4.7 and the graph on Figure 4.8 provides the velocity of the valve's master node versus time.

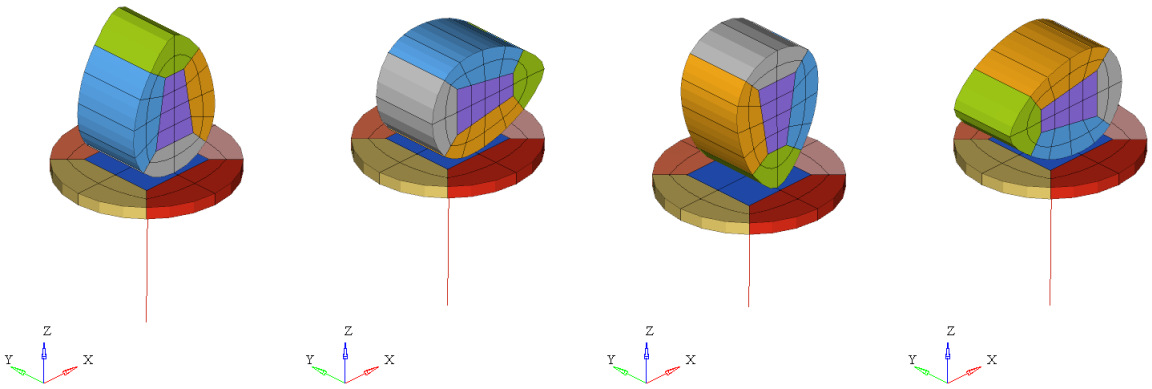


Fig. 2.23: Cam and valve positions during the simulation.

The raw results obtained are noisy due to the fact that isogeometric elements are linearized for contact algorithms and the penalty method applies discontinuous forces. The oscillations correspond to each application of a contact force to the valve. It can be noticed that these oscillations have a similar amplitude between the two models, even if the NURBS element density is lower than the FE one. Indeed, despite the fact that the size of the finite elements on the circumference of the cam is smaller than the coarse NURBS one, the linearization of the

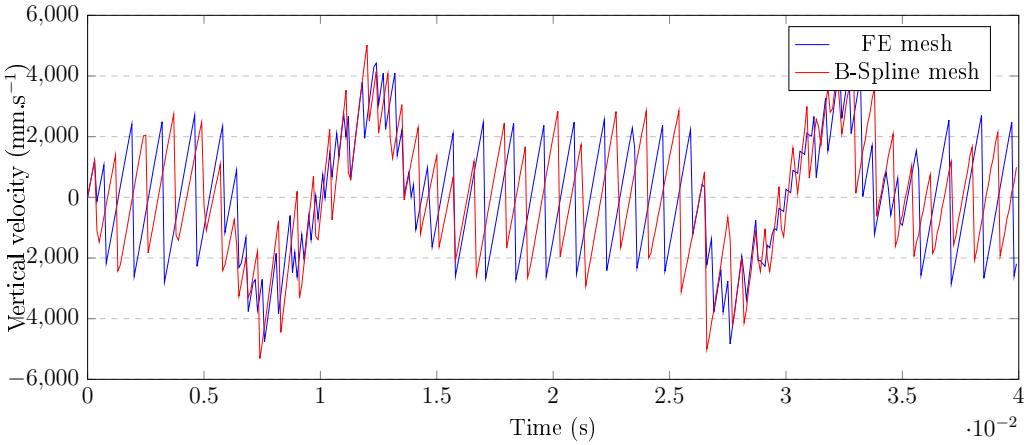


Fig. 2.24: Valve's master node vertical velocity.

quadratic elements NURBS multiplies the number of contact segments by 3 per direction. It makes the NURBS contact surface discretization almost as fine as FE one.

A smooth velocity curve, see Figure 4.9, is obtained by using a low pass filter. The filtering quality depends on the number of samples which in this case is the number of points computed by Radioss for each curve. We can see that despite some small variations, the behavior is very similar.

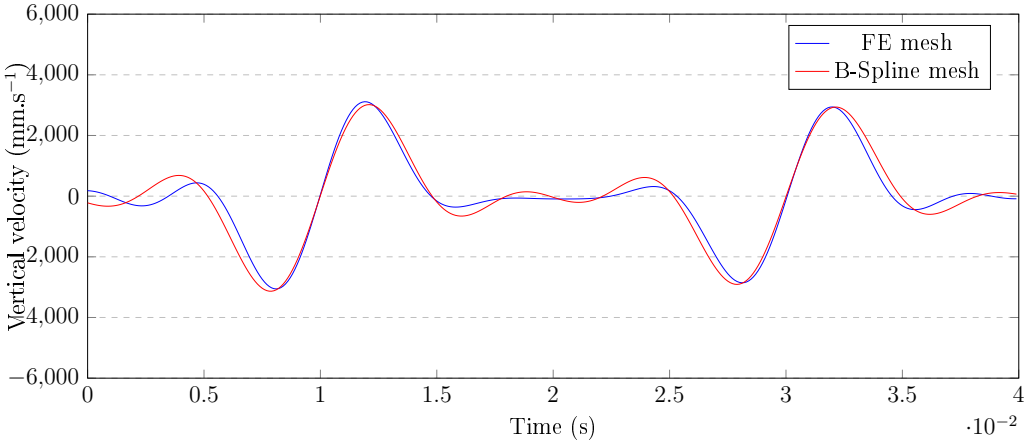


Fig. 2.25: Filtered valve's master node vertical velocity.

3D Local refinement: Locally Refined B-Splines implementation

Contents

In the industrial field, mesh engineers who use explicit industrial codes are used to follow mesh size recommendations that depend on the part and the type of simulation to be performed. They take into account the geometric non-linearities of the parts to be meshed, the boundary condition application areas, etc. Generally, these instructions are given by the office of the methods. It is common to find 5 mm or 2.5 mm size meshes, and models are often discretized in a uniform way, see for example Figure 3.1. Despite the fact that it provides quality results, these meshes are very fine, very costly and require the use of parallelism. Car crash simulations are a good illustration of this.

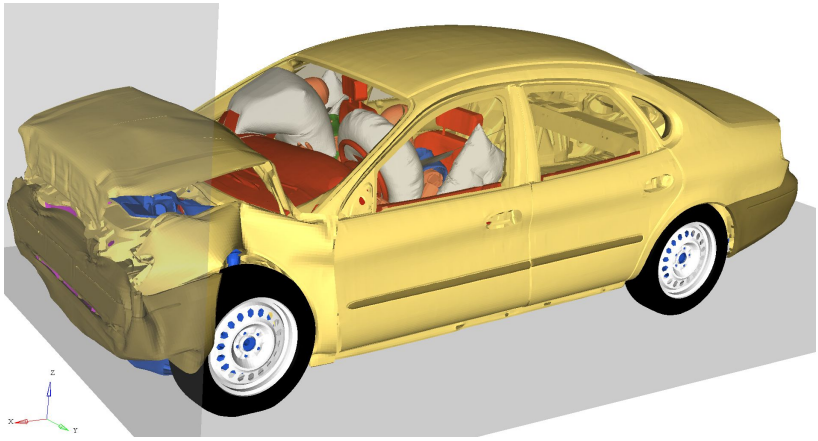


Fig. 3.1: Simulation result of an entire car crash with dummies. The model is composed of 10 millions elements with a mesh size of 2.5 mm.

Other simulations can use adaptive meshes where we can have several mesh sizes, hierarchical meshes and so transition areas. This is quite often the case for stamping simulations where the mesh size adapts according to the areas which are strongly loaded or distorded. Obtained results are good, but these methods are quite complex to implement in industrial codes with classical architecture and are also costly.

The classical B-Spline based IGA has been implemented in several implicit and explicit codes and showed accuracy and robustness on academic cases. The use of B-Splines for industrial cases is however less easy since the associated global tensor product prevents local refinement when it is needed, see Figure 3.2. The challenge of local refinement methods for B-Splines is then to introduce new types of basis functions, which always allow to apply the minimum mesh size recommendations in non-linearity areas and to have a bigger mesh size elsewhere. Transition areas are therefore important in explicit simulations since they avoid elastic wave return phenomena. These local refinement methods implemented within Radioss are not adaptive, they therefore require the experience of the mesh engineer.

After comparing several Spline basis functions allowing local refinement, the work aims to implement the LR B-Spline technology in Radioss explicit code. The existing structures have in their center the element are not really adapted to the integration of a formulation which has the patch or the control point as central point. Some modifications or implementation choices had consequently to be done in order to perform it use in Radioss.

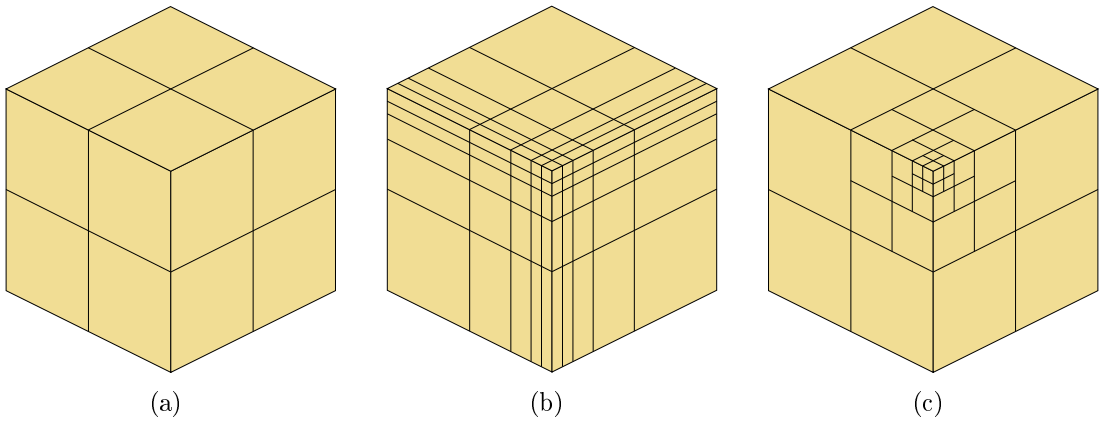


Fig. 3.2: Illustration of the local refinement issue for B-Splines: (a) initial mesh, (b) tensor product refinement, (c) targeted local refinement.

3.1 Brief comparison of Spline functions for an implementation in an explicit solver

Spline functions can be summarized and compared taking into account all the characteristic needed for their implementation in an explicit code, i.e. linear independence, partition of unity, no overloaded elements, available 3D implementation, implementation constraints within Radioss, see Table 3.1.

The choice of the locally refined functions has been greatly influenced by the fact that according to the functions it is possible for one element to have more than $(p + 1)$ control points per direction with p the degree polynomial function. Structures related to existing elements in a finite element code are usually designed and used for a fixed number of "nodes". This number of "nodes" depends on the element type and the polynomial degree of its shape functions but is never changed. The Hierarchical and Truncated B-Splines were dismissed in part because of this consideration. The main reason for the search for this minimal number of functions being initially guided by the need to maintain

	HB-Splines	THB-Splines	LRB-Splines
Partition of unity	✓/ ×	✓	✓
Linear independence	✓	✓	✓/ ×
No overloaded elements	×	×	✓/ ×
Implementation constraints (no nested structure)	×	×	✓

Table 3.1: Comparison of Spline technologies.

linear independence, it is all the more important in terms of integration into already existing data structures. LR B-Splines can ensure this constraint under conditions and has been chosen as the best compromise to implement inside Radioss.

3.2 Locally Refined B-Splines

Locally Refined B-Splines, introduced by Dokken et al. [37], allow to avoid the NURBS and B-Splines global tensor product constraints and permit for the construction of a basis on a more general mesh which may include T-junctions.

3.2.1 Definition

As presented earlier, LR B-Splines use a local knot insertion for local refinement. In 1D, inserting one new knot $\hat{\xi}$ in a local knot vector Ξ_i of a B-Spline function B_{Ξ_i} will split the function into two new ones B_{Ξ_1} and B_{Ξ_2} depending the following expression:

$$B_{\Xi_i}(\xi) = \alpha_1 B_{\Xi_1}(\xi) + \alpha_2 B_{\Xi_2}(\xi), \tag{3.1}$$

The local knot vector Ξ_i is then split in two local knot vectors Ξ_1 and Ξ_2, with the same length depending the polynomial degree of B_{Ξ_i}:

$$\begin{aligned} \Xi &= [\xi_1, \xi_2, ..., \xi_{i-1}, \ \xi_i, ..., \xi_{p+1}, \xi_{p+2}], \\ \Xi_1 &= [\xi_1, \xi_2, ..., \xi_{i-1}, \hat{\xi}, \xi_i, ..., \xi_{p+1} \quad\ ], \\ \Xi_2 &= [\quad \xi_2, ..., \xi_{i-1}, \hat{\xi}, \xi_i, ..., \xi_{p+1}, \xi_{p+2}]. \end{aligned} \tag{3.2}$$

The coefficients α_1 and α_2 are given as follows:

$$\alpha_1 = \begin{cases} \frac{\hat{\xi}-\xi_1}{\xi_{p+1}-\xi_1} & \text{if} \quad \xi_1 \leq \hat{\xi} \leq \xi_{p+1} \\ 1 & \text{if} \quad \xi_{p+1} \leq \hat{\xi} \leq \xi_{p+2} \end{cases}, \qquad \alpha_2 = \begin{cases} 1 & \text{if} \quad \xi_1 \leq \hat{\xi} \leq \xi_2 \\ \frac{\xi_{p+2}-\hat{\xi}}{\xi_{p+2}-\xi_2} & \text{if} \quad \xi_2 \leq \hat{\xi} \leq \xi_{p+2} \end{cases}. \tag{3.3}$$

A set of quadratic B-Spline basis functions with a global knot vector $\Xi = [0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8]$ is used to illustrate this. The basis is composed of 10 functions with 10 associated local knot vectors. At the beginning, each basis function has a defined scaling weight $\gamma = 1$. A new knot $\widehat{\xi}$ is to be inserted in this basis, see Figure 3.3.

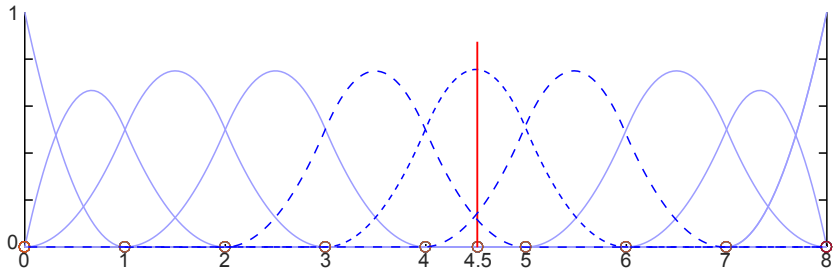


Fig. 3.3: Knot insertion for $\widehat{\xi} = 4.5$: Dashed curves is the original functions to be split. Light blue functions will not be modified since the new knot is outside of their support.

The new knot $\widehat{\xi}$ will split 3 existing basis functions, plotted in dotted blue line in Figure 3.3. The split and the calculation of α_1 and α_2 for each of these functions has to be done independently, see Figure 3.4. The associated scaling weight of the new fonctions is then calculated following:

$$\gamma_1 = \alpha_1 \gamma_i \quad \text{and} \quad \gamma_2 = \alpha_2 \gamma_i, \tag{3.4}$$

respectively for the left and the right new basis functions.

Then, the resulting functions B_{Ξ_1} and B_{Ξ_2} have to be replaced in the entire basis. If one of the two resulting functions is not already present in the basis, it is simply inserted. If it already exists, due to the split of a contiguous function and a previous insertion of a new one, it will modify the weight γ according to the relation:

$$\gamma_{existing} = \gamma_{existing} + \alpha_i \gamma_i. \tag{3.5}$$

The resulting basis is given in Figure 3.5

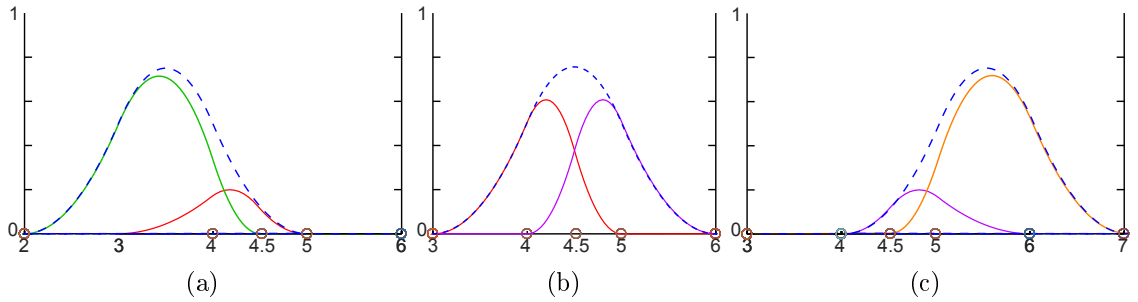


Fig. 3.4: Knot insertion for $\widehat{\xi} = 4.5$ in the 3 dashed B-Spline basis functions of Figure 3.3. The colored ones are the resulting functions. Notice that new B-Spline functions that have the same support are in the same color.

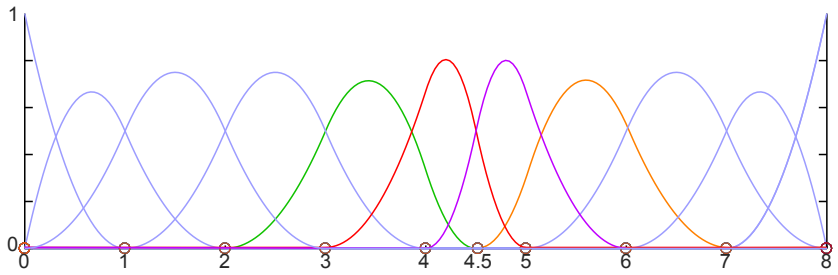


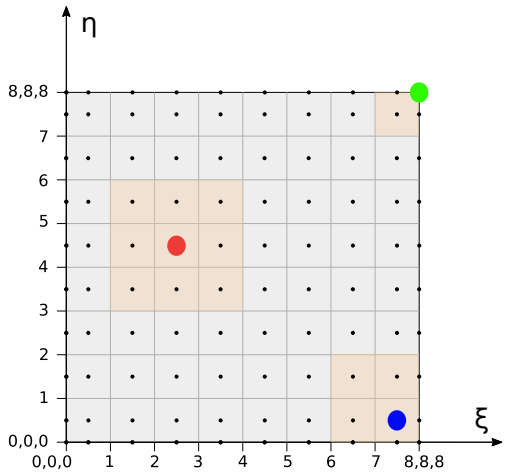
Fig. 3.5: Result of knot insertion for $\xi = 4.5$ in the quadratic basis.

3.2.2 Mesh lines and 2D refinements

In 2D, a local knot vector is defined for each B-Spline basis function with an extent of $[p+1] \times [q+1]$ where p and q are the polynomial degrees by direction. An illustration of some 2D quadratic basis is given in Figure 3.6.

The split of basis functions in the patch in 2D is done by inserting a so-called mesh line. A mesh line is defined by a specific knot value $\widehat{\xi}$ or $\widehat{\eta}$ to be inserted in one of the two directions and by an interval in the other direction $[\eta_1, \eta_2]$ or $[\xi_1, \xi_2]$. This knot interval corresponds to the mesh line length. The specific knot value can be assimilated as the inserted knot value in 1D cases. A given mesh line is said to traverse a B-Spline B_i if it passes through all of its support, see Figure 3.7(a). Only in that case the mesh line will split one or several B-Spline functions. As a comparison, we can see in Figure 3.7(b) another meshline that is not extended enough to split any B-Spline function.

One B-Spline B_i can thus be split at the knot $\widehat{\xi}$ or $\widehat{\eta}$ and will also introduce two new B-Splines B_1 and B_2 like in 1D, in one of the parametric direction. The replacement of the two new B-Splines in the existing space will update them if they already exist, i.e., if there is already a B-Spline function with the same local knots.



i	Ξ_i	$\times$	H_i
1	$[1,2,3,4]$	$\times$	$[3,4,5,6]$
2	$[6,7,8,8]$	$\times$	$[0,0,1,2]$
3	$[7,8,8,8]$	$\times$	$[7,8,8,8]$

Fig. 3.6: B-Spline basis function support for a 2D quadratic patch, colored in light orange with the corresponding control point $\boldsymbol{C_i}$ in the parametric space and local knot vectors for the three arbitrary given B-Splines.

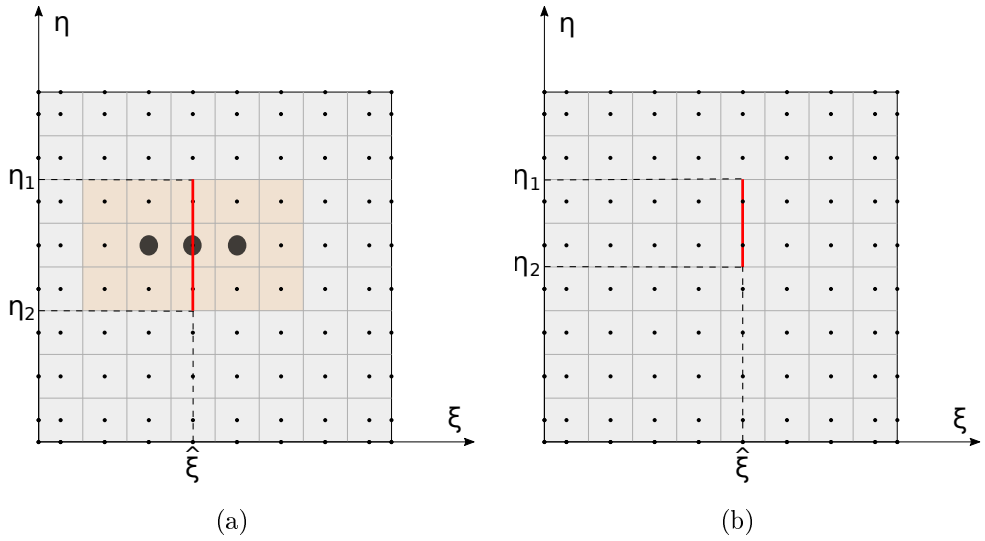


Fig. 3.7: A mesh line is defined by a specific knot value $\hat{\xi}$ and by a span in the other parametric direction $[\eta_1, \eta_2]$. a) The mesh line traverses entirely the support of a set of 3 B-Splines at $\widehat{\xi}$, b) The mesh line is not extended enough and will not split any functions.

It will create new ones if not.

In a 2D space of functions we realize that the use of local knot vectors rather than global vector allows to overcome the global tensor product. Each function has a completely independent support compared to its neighbors and can be split

by inserting a mesh line. The length of the mesh line is then arbitrary and the refinement is no longer extended to the edges of the patch.

Crossed refinements Since the models are no longer unidirectional, we can end up in cases of multiple and potentially crossed refinements, i.e., with intersecting meshlines. Inserting a meshline at a given knot value will create or modify functions that can in turn be split by existing meshlines in the other parametric direction. Indeed, these B-Spline functions see their local knots modified by the first meshline and can therefore be entirely traversed by other old meshlines. This is called "cross-checks" of functions, see Figure 3.8.

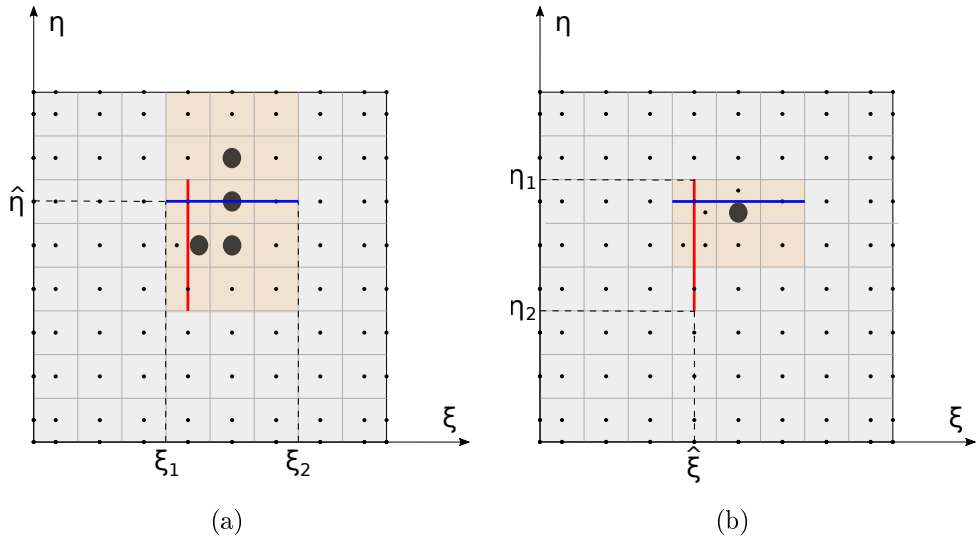


Fig. 3.8: Crossed refinement: a) The blue mesh line traverses entirely the support of 4 functions at $\widehat{\eta}$ b) One of the resulting functions of the blue mesh line is cut by the existing red mesh line of Figure 3.7(a)

3.2.3 Extension to 3D refinement

Even if LR B-Splines were initially introduced for uni-variate and bi-variate cases [37], this technology can be extended to the trivariate one. Some informations were given in appendix in [57] about this extension. As explained earlier, there is no particularity to the 3D usage of LR B-Splines and the basis functions can also be given with local knot vectors and their support can be split in the same way. This allows the mesh line extension to higher dimension.

In 3D, refinements are generated using mesh surfaces which are the counterparts of mesh lines in 2D. A mesh surface is defined by a specific knot value $\widehat{\xi}$, $\widehat{\eta}$ or

$\widehat{\zeta}$ which gives a normal direction, and by a bounded surface by knot coordinates in the other two directions. Once again, one or several B-Splines B_i which are traversed entirely by a mesh surface will be split. A very simple quadratic 3D case, where the mesh surface is a rectangular surface, is illustrated in Figure 3.9. It shows the mesh surface components, i.e. the new knot $\widehat{\xi}$ and the rectangular surface $[\eta_1, \eta_2] \times [\zeta_1, \zeta_2]$, before the split of the B-Spline functions. The black functions with a support that is completely traversed by the mesh surface will be split. For better understanding, a simplified way to see a mesh surface is to project it in a parametric plane at the depth of the inserted new knot, i.e. on the (η, ζ) plane at the depth $\widehat{\xi}$, at the bottom left in Figure 3.9. In 3D, more than in 2D, the number of crossed refinements can be important and there may be at each mesh surface insertion of a large number of these secondary splits in each other parametric direction.

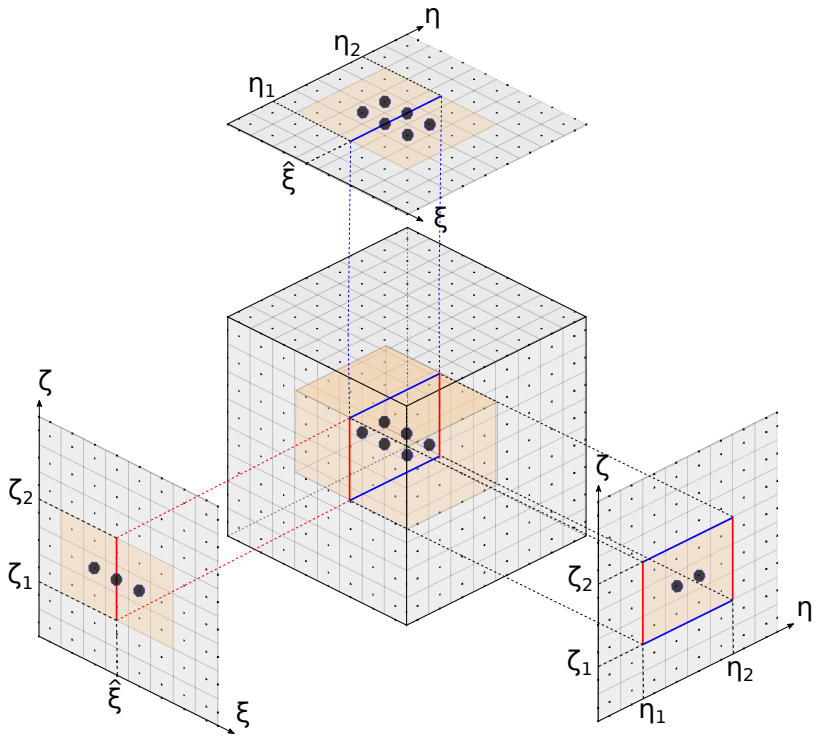


Fig. 3.9: Mesh surface is defined by a normal direction with a specific knot value $\hat{\xi}$ and by a span in the other parametric direction $[\eta_1, \eta_2]$ and $[\zeta_1, \zeta_2]$. The mesh surface is projected on planes (ξ, η), (η, ζ) and (ζ, ξ) for a better visualization.

3.3 The linear independence issue

A LR B-Spline basis is said to be linearly independent if it ensures the following:

$$\sum_{i=1}^{n} c_i B_i(\xi,\eta,\zeta) = 0, \quad \forall(\xi,\eta,\zeta) \in R \Rightarrow c_i = 0 \forall i, \tag{3.6}$$

with $i = 1, n$ the set of B-Splines of the base, depending on the dimension of the base. In 1D, a knot insertion shifts the bounds of the local knots of the LR B-Spline functions and the constructed LR B-Spline base is linearly independent. In 2D and 3D, the knot insertion is a little bit more complex and this property is not guaranted, as it will be illustrated for quadratic meshes.

Based on the work of Bressan [23], Bressan and Jüttler [24], we will consider that a sufficient condition to guaranty the linear independence property is to have no overloading. As a reminder, a trivariate element $R = [\xi_1, \xi_2] \times [\eta_1, \eta_2] \times [\zeta_1, \zeta_2]$ of polynomial degree p, q and r will be said overloaded if there are more than $(p+1)(q+1)(r+1)$ non-zero B-Spline functions with support on R.

The linear independence and the non-overloading aspects are important in terms of implementation and the subsequent use in the data structures of a solver, as will be shown later.

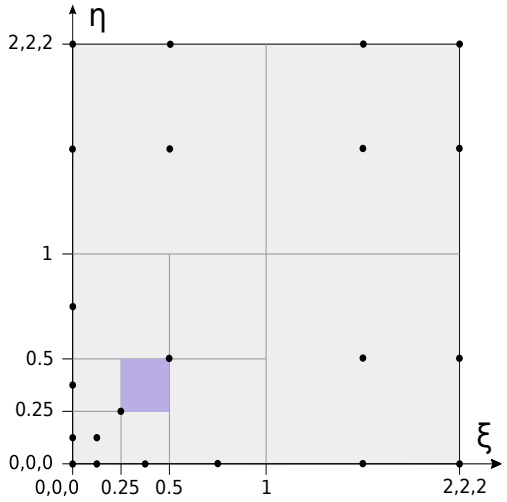
3.3.1 Analysis of 2D refined mesh

A 2D similar refinement to the refinement pattern given of Figure 3.2(c) is defined in the (ξ, η) plane with the corresponding knot vectors. It is studied for a quadratic basis. All non-zero B-Spline functions for two elements are listed in Figure 3.10 and Figure 3.11 and show overloading in this quadratic patch. Notice that for the second overloaded element, the number of function is 13, whereas it should be 9.

This 2D refinement scheme, although it is often used to illustrate local refinement, is not suitable for LR B-Splines. Moreover, in the context of an explicit finite element code, it is customary to have refinements that are established in a progressive manner. To have a refinement too sudden risks to cause a rebound of elastic waves because of the discontinuity of the mesh. In the case studied here, the refinement is too abrupt and is similar to hierarchical refinements, with too small transition areas.

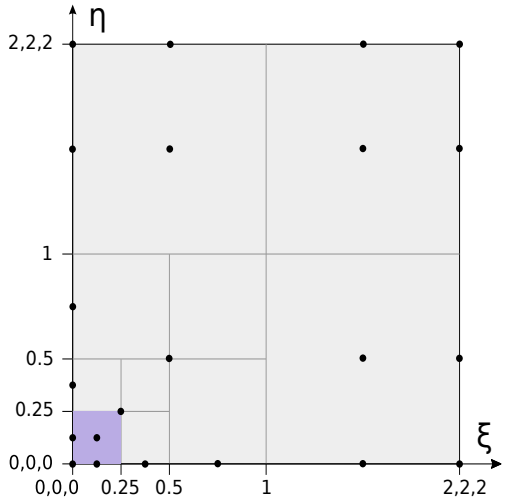
3.3.2 Improved full span refinement scheme

Johannessen et al. [57] addressed the overloading and the linear independence loss problem through three refinement schemes, called min span, structured mesh and full span. Whereas the first two ones give overloaded elements in very simple cases,



i	Ξ_i	×	H_i
1	$[0, 0, 0.25, 0.5]$	×	$[0, 0, 0.25, 0.5]$
2	$[0, 0, 0.5, 1]$	×	$[0, 0, 0.5, 1]$
3	$[0, 0.5, 1, 2]$	×	$[0, 0, 0, 0.5]$
4	$[0, 0, 0, 0.5]$	×	$[0, 0.5, 1, 2]$
5	$[0, 0.5, 1, 2]$	×	$[0, 0, 0, 1]$
6	$[0, 0, 0, 1]$	×	$[0, 0.5, 1, 2]$
7	$[0, 0, 1, 2]$	×	$[0, 0, 1, 2]$
8	$[0, 1, 2, 2]$	×	$[0, 0, 1, 2]$
9	$[0, 0, 1, 2]$	×	$[0, 1, 2, 2]$
10	$[0, 1, 2, 2]$	×	$[0, 1, 2, 2]$

Fig. 3.10: Non-zero LR B-Splines on $[0.25, 0.5] \times [0.25, 0.5]$. This element is overloaded.



i	Ξ_i	×	H_i
1	$[0, 0, 0, 0.25]$	×	$[0, 0, 0, 0.25]$
2	$[0, 0, 0.25, 0.5]$	×	$[0, 0, 0.25]$
3	$[0, 0, 0, 0.25]$	×	$[0, 0, 0.25, 0.5]$
4	$[0, 0, 0.25, 0.5]$	×	$[0, 0, 0.25, 0.5]$
5	$[0, 0, 0.5, 1]$	×	$[0, 0, 0.5, 1]$
6	$[0, 0.5, 1, 2]$	×	$[0, 0, 0, 0.5]$
7	$[0, 0, 0, 0.5]$	×	$[0, 0.5, 1, 2]$
8	$[0, 0.5, 1, 2]$	×	$[0, 0, 0, 1]$
9	$[0, 0, 0, 1]$	×	$[0, 0.5, 1, 2]$
10	$[0, 0, 1, 2]$	×	$[0, 0, 1, 2]$
11	$[0, 1, 2, 2]$	×	$[0, 0, 1, 2]$
12	$[0, 0, 1, 2]$	×	$[0, 1, 2, 2]$
13	$[0, 1, 2, 2]$	×	$[0, 1, 2, 2]$

Fig. 3.11: Non-zero LR B-Splines on $[0, 0.25] \times [0, 0.25]$. This element is also overloaded.

the full span scheme can be used but only for rectangular refinements.

A 2D refinement of a L-shaped domain is given as an example of non-rectangular refinement. The refinement of the left and the lower parts of a 2D mesh forms an L, see Figure 3.12 for a quadratic basis. In this case, the corner of the L is a re-entrant one and the full span scheme does not avoid the presence of overloaded elements. The set of overloaded elements is highlighted in blue in Figure 3.12. This overloading is partly due to the too large extent of the highlighted quadratic

function and to too many refined elements in the corner.

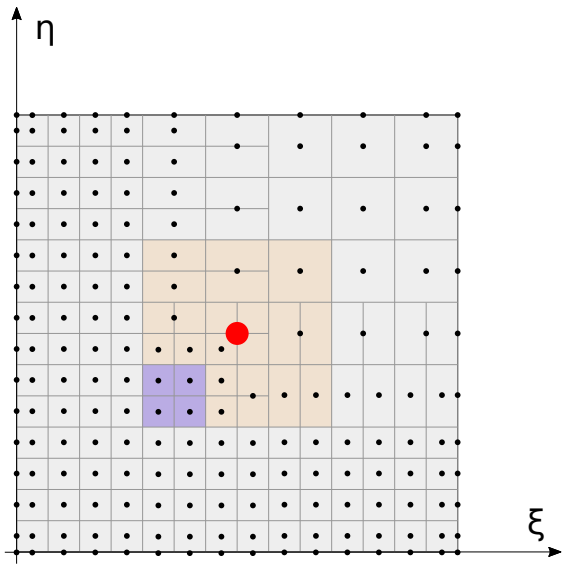


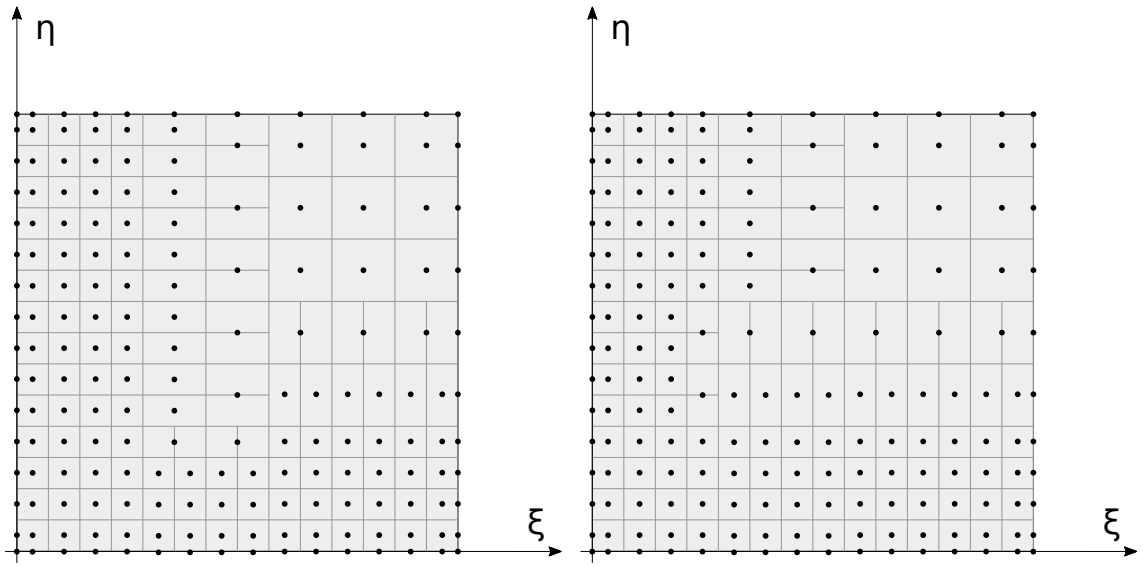
Fig. 3.12: L-shaped refinement of a 2D basis of quadratic functions losing the linear independence property. The red function partly cause tue overloading of gray elements.

The solution to recover a non overloaded basis is to discard the refinement in one direction for the first p elements near the corner. We have two solutions depending on the order in which we refine the parametric directions, see Figure 3.13. In this way, one comes to reduce the number of refined elements in the re-entrant corner but it is no longer overloaded.

Several working refinement shapes can be obtained with this scheme, like cross-shaped refinement, stair-shaped refinement or refinement with re-entrant corner, and the absence of overloaded elements is ensured. In conclusion, we conjecture that with this choice of the full span and the improved full span schemes we will always be in a subset of the linearly independent LR B-Spline basis without overloading which are analysis suitable.

3.3.3 Extension to 3D refinements

The same linear independence and overloaded elements issue is found for 3D cases. The higher complexity of the trivariate case and the possibly secondary splits obtained after a mesh surface insertion might induce that some of the element of the resulting mesh can be overloaded, with random shaped refinements.



(a) (b)

Fig. 3.13: Modification of the refinement of p elements by direction of the entering corner, see Figure 3.12, to keep the non overloaded mesh. These elements are now refined in one direction. The number of new functions introduced is reduced and avoids their overlapping. Illustration of the vertical refinement for Figure 3.13(a) and horizontal for Figure 3.13(b).

As said earlier, regular 3D models are not more than extension in the third parametric direction of 2D regular models. The scheme of the regular tridimentionnal mesh can then be resumed as the corresponding 2D scheme. Keeping in mind that point for regular models, it comes that a 3D refined one can be seen as a set of 2D models at several given knot depths, and this is true for all the three parametric directions. It means that a corresponding planar mesh can be found each time there is a knot value in the global or local knot vectors for the normal direction.

Even if a direct extension of the full span or the improved full span refinement scheme stays possible, it would be very complex. So, for 3D LR B-Spline refined model, these schemes have to be applied on each 2D slice of the three directions of the model, at each knot depth.

An example of 3D refined model with several mesh sizes, giving a progressive and tridirectionnal refinement is given in Figure 3.14. Some of the corresponding 2D slices can be extracted and show full span schemes. It can be seen that the refinement scheme is applied at several levels to obtain the smoothness of the refinement, giving an analysis-suitable model.

Another 3D refined model is given in Figure 3.15, showing this time the improved full span scheme application. In this refinement, several levels of the improved full span are used, also in order to ensure the analysis-suitability of the model. We can notice that if we project the 2D models corresponding to the outer faces, we find full span schemes.

3.4 3D LR B-Splines implementation in Radioss

LR B-Splines were applied to IGA by Johannessen et al. [57, 58], but, to the best of our knowledge, not in an explicit software. Their characteristics are however adapted to their use and their 3D extension capability proposed in [57] is particularly interesting for crash simulations using solid models. However, the use of local refinement methods implies having a preprocessing software capable of generating the appropriate meshes and parameterizations, like classical IGA. It is also necessary that the preprocessor has the ability to check the important properties for the analysis as the no overloading condition for the case of LR B-Splines. To this date and to our knowledge, there is no such tool available.

The 3D LR B-Splines have been implemented inside the Radioss solver considering that the refined model has been previously defined in a unique and static way. Radioss solver has to interpret the refinement instructions, possibly to verify that the result is analysis suitable, and to refine the regular model of the input file.

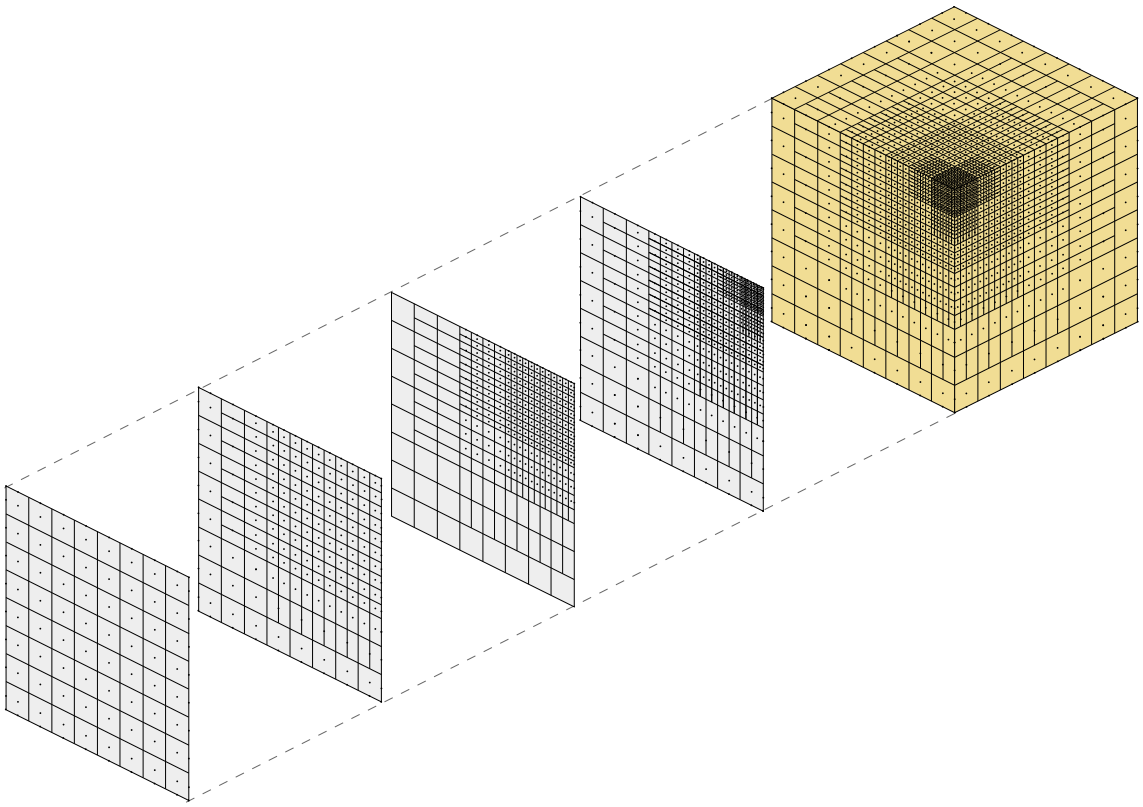


Fig. 3.14: Refinement applied following the pattern of full span on all three sides and inside a cube. There is no overloaded element. The gray planes are the different sections illustrating the diferent levels of refinement in one of the three directions.

3.4.1 Refinement instructions

The same element property introduced in the previous chapter is used to defined locally refined elements and the refinement instructions are given by the pre-processor for each initialy regular isogeometric element. Each element can be cut in 2^{L_i} finer elements, with L an arbitrary number defining the minimal mesh size and i the parametric direction.

Defining the refinement in that way, we keep a finite element point of view in the input file and let the basis functions and their split to be treated inside Radioss. There is no need to define local knots in the input file for example. The global knot vectors and the element connectivity are still given in the data file as an unrefined model. It is a way of limiting the amount of information that has to be given through the input file and facilitates the user usage. The input file documentation for the locally refined isogeometric element will be available in the Radioss online help and is included in the appendix A.

All other commonly used data such as global knot vectors by direction or element

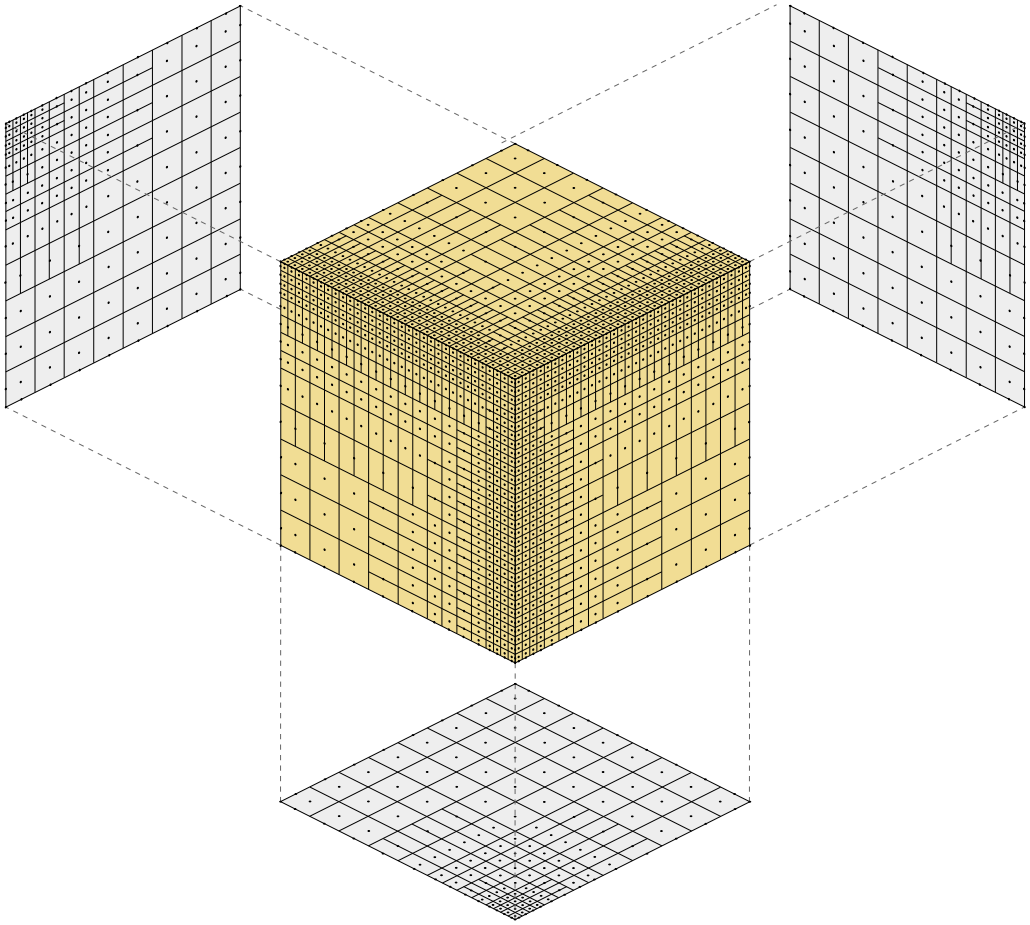


Fig. 3.15: L-shaped refinement taking into account the remark on the re-entrant corners refinement. There is no overloaded element. Grey planes show hided faces where can be retrieved a full span pattern on all three sides.

connectivity do not change with respect to an unrefined model.

3.4.2 Unit mesh surface creation and basis functions sorting

The case of a unit mesh surface creation is presented here for simplicity and illustration purpose. In the general case, a refined model will use several mesh surfaces.

3.4.2.1 Mesh surface creation

The rectangular mesh surface, introduced in Figure 3.9, is the 3D extension of the meshline and is defined by two extended knots [57]. Its generalization in 3D can be widened by defining it by a set of parametric coordinates forming a non-secant polygon, see Figure 3.16. With this generalization, we limit the total number of mesh surfaces compared to its previous definition for the same refined

mesh since there is no need to group the rectangular mesh surfaces that are adjacent.

Mesh surfaces are built by grouping elements to be refined in the same way, see an example in Figure 3.16(b) at a given $\widehat{\xi}$ depth. From the element list, description of the elements of the same parametric plane, a normal parametric value $\widehat{\xi}$ is defined as the new knot to be inserted. Then a surfboard is defined from this set of elements and from $\widehat{\xi}$.

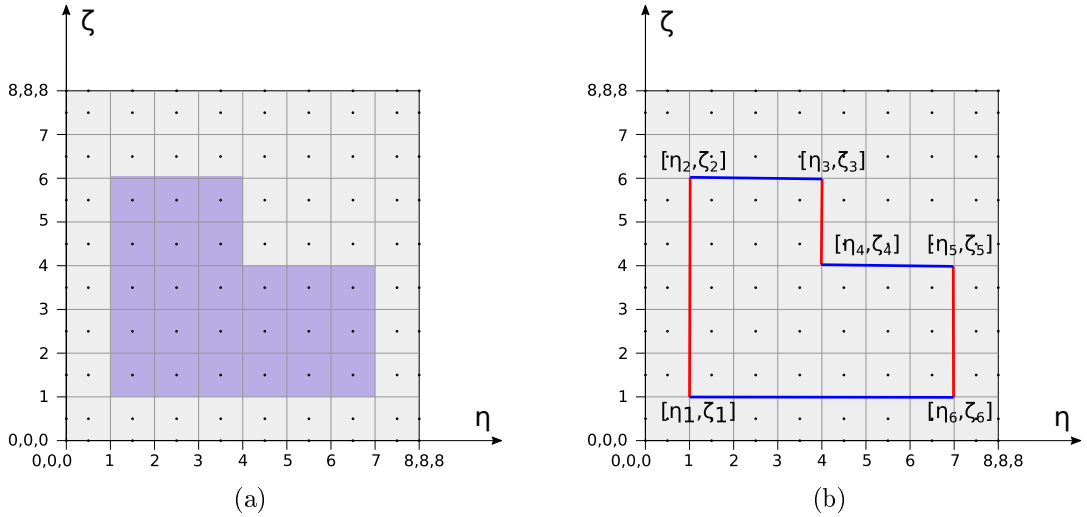


Fig. 3.16: Arbitrary mesh surface creation from element refinement instructions given by the user in the input data file. a) Neighbors elements from the same parametric plane which are to be refined in a similar way. b) Resulting mesh surface contour.

We can notice that we do not have the case of a mesh surface extension as described in [57]. Indeed, this particular mesh surface definition avoids having to take into account this aspect.

3.4.2.2 Sorting of functions to be split

The sorting of functions cut by the mesh surface is not done on all the model, as there would be too many functions to test. The connectivity tables of the selected elements give a list of functions in which one can be sure that all functions to be refined are there. These are necessarily functions that will be non-zero on the selected elements. The functions of Figure 3.16 to be refined are given in Figure 3.17.

This mesh surface creation and basis function selection split the minimal number of shape functions that results in the refinement of the elements selected

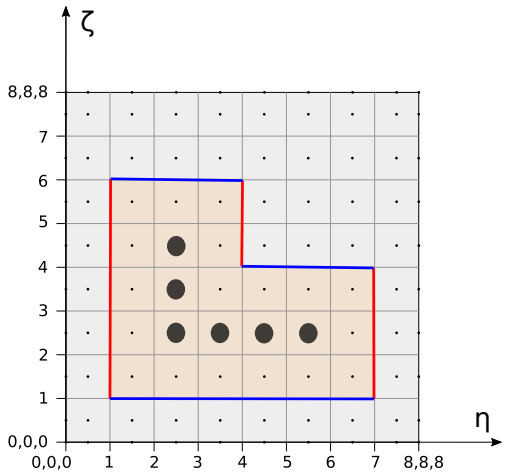


Fig. 3.17: Neighbors B-Spline functions which are cut by the mesh surface of Figure 3.16(b) are set to be split.

by the user. However, if the user does not provide enough elements or an element area that results in a mesh surface that does not completely cut any function, the element will not be refined.

Figure 3.18 gives the 3D illustration of the defined mesh surface at a given $\widehat{\xi}$ depth. We can see on each direction the mesh surface projection and all the functions to split.

3.4.2.3 Refinement algorithm

Johannessen et al. [57] give the split algorithm and refinement processus, for the 2D case. It is generalized to be used in 3D in this work. The procedure implemented in Radioss can be summarized in 3 steps:

- Considering the element point of view, group the information of the elements to be refined, i.e., parametric direction and number of refinement. Group these elements and construct the mesh surface that corresponds to the refinement of this element or group of elements.
- Depending on the local knot vectors that are stored in memory, find and split any B-Spline whose support is completely cut by the new mesh surface, see Algorithm (2) and Algorithm (3).
- For all new basis function split, check if their support is completely traversed by any existing mesh surface. In this case, split it again according to this

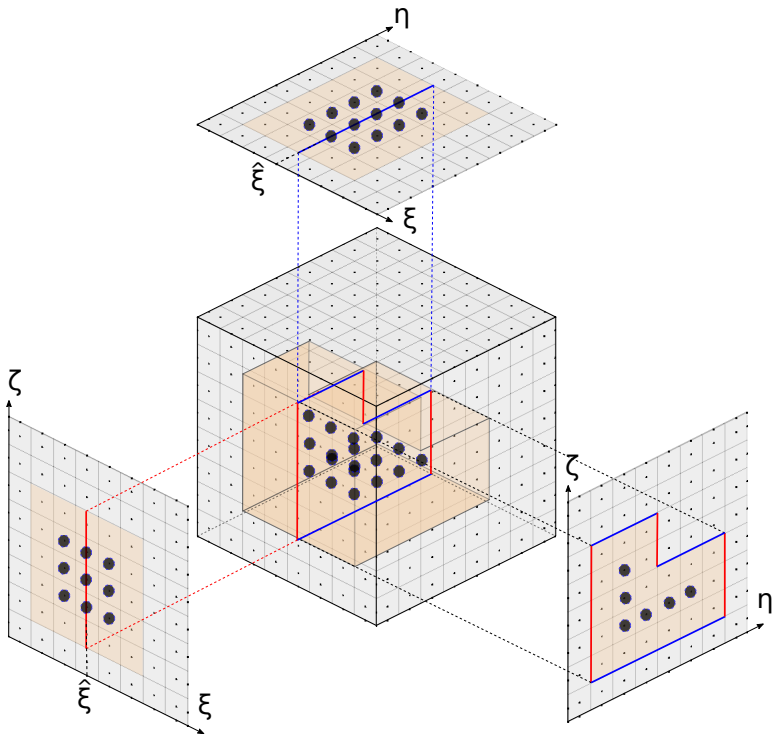


Fig. 3.18: Arbitrary-shaped mesh surface from Figure 3.16(b) inside the 3D mesh. 3D mesh with projection of mesh surface and B-Spline functions in each plane. Bold functions are to be split.

existing mesh surface.

with $\underline{c_i}$ the spatial coordinates of the i^{th} control point. This algorithm is very similar to the one introduced in Johannessen et al. [57]. However, the complexity of a model and a 3D refinement induces that there are many more basis function modifications by existing mesh surfaces than for 2D cases.

3.4.3 Illustration on a multi mesh surfaces case

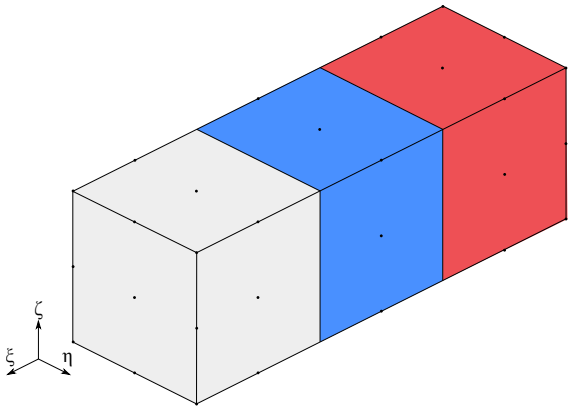
Let us take now the case of a quadratic B-Spline patch composed of 3 elements to be refined several times. Refinement instructions are given in the input file to refine one of the elements quite finely in all three directions. The refinement of the neighboring element is defined to obtain a transition area between the fine level and the coarse level. Usually, the notion of level refers to Hierarchical B-Spline function levels. In our work, what is called refinement level corresponds to an area of the mesh which has a given size of elements.

Algorithm 2 3D LR B-Spline refinement, from [57]

parameters: S {Spline space}, M {LR mesh}, ϵ {Mesh surface}

for every B-Spline $B_i \in S$ **do**
  if ϵ splits B_i **then**
    perform split according to Algorithm (3)
  end if
end for
for every B-Spline $B_i \in S_{new}$ **do**
  for every existing edge $\epsilon_j \in M$ **do**
    if ϵ_j splits B_i **then**
      perform split according to Algorithm (3)
      (Note that this may enlarge S_{new} further).
    end if
  end for
end for

The initial mesh and the refinement instructions are given in Figure 3.19.



Element	Number of cut in ξ	η	ζ
1	4	4	4
2	2	2	2
3	1	1	1

Fig. 3.19: Initial mesh and refinement instructions.

Following the method set out earlier, a set of mesh surfaces equivalent to the refinement of elements can be defined for each of the three directions. They are given by parametric direction in the figure Figure 3.20.

We can also note that the red surface mesh of the figure Figure 3.20 does not only cut the first element, but also the second one.

With the set of defined mesh surfaces, sorting and splitting B-Spline functions can be done. As mentioned in [57], this process has to be done one mesh surface

Algorithm 3 Local ξ-split for the 3D case, from [57]

parameters: $\widehat{\xi}$ {New knot}; B_i {B-Spline to be split ($B_i \in S$)}; S {Spline space}; S_{new} {Functions not present in S}

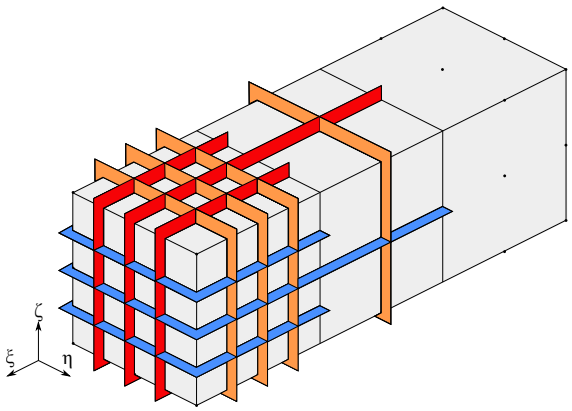
calculate (α_1, α_2) from Equation (3.3)
$\Xi \leftarrow SORT(\Xi \cup \widehat{\xi})$
$\Xi_1 \leftarrow [\xi_1, ..., \xi_{p+2}]$
$\Xi_2 \leftarrow [\xi_2, ..., \xi_{p+3}]$
$\Psi_1 \leftarrow \Psi_i$
$\Psi_2 \leftarrow \Psi_i$
$Z_1 \leftarrow Z_i$
$Z_2 \leftarrow Z_i$
if $(\Xi_1, \Psi_1, Z_1) \in S$ **then**
    $\underline{c_1} \leftarrow (\underline{c_1}\gamma_1 + \underline{c_i}\gamma_i\alpha_1)/(\gamma_1 + \alpha_1\gamma_i)$
    $\gamma_1 \leftarrow \gamma_1 + \alpha_1\gamma_i$
else
    $\underline{c_1} \leftarrow \underline{c_i}$
    $\gamma_1 \leftarrow \alpha_1\gamma_i$
    add B_1 to S_{new}
end if
if $(\Xi_2, \Psi_2, Z_2) \in S$ **then**
    $\underline{c_2} \leftarrow (\underline{c_2}\gamma_2 + \underline{c_i}\gamma_i\alpha_2)/(\gamma_2 + \alpha_2\gamma_i)$
    $\gamma_2 \leftarrow \gamma_2 + \alpha_2\gamma_i$
else
    $\underline{c_2} \leftarrow \underline{c_i}$
    $\gamma_2 \leftarrow \alpha_2\gamma_i$
    add B_2 to S_{new}
end if
remove B_i from S.

after the other, to be able to split all B-Spline functions for the actual mesh surface, but also to find all secondary splits with old mesh surfaces.

The resulting refined mesh is given in Figure 3.21. The initial colors of element are used for the corresponding refined elements.

3.4.4 Transition areas and refinement extents

The refinement is based on a regular initial mesh, considering that elements can be refined according to the powers of 2. This constraint implies that refinements have a given extent. Let's take as an example the refined mesh of Figure 3.2(c) and



Normal Direction	Number of mesh surfaces
ξ	4
η	3
ζ	3

Fig. 3.20: Equivalent mesh surfaces corresponding to the refinement instructions.

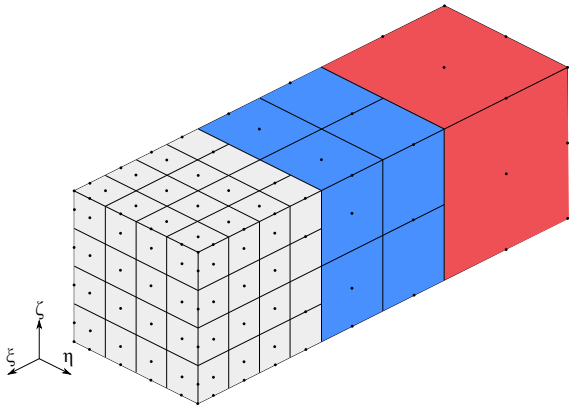
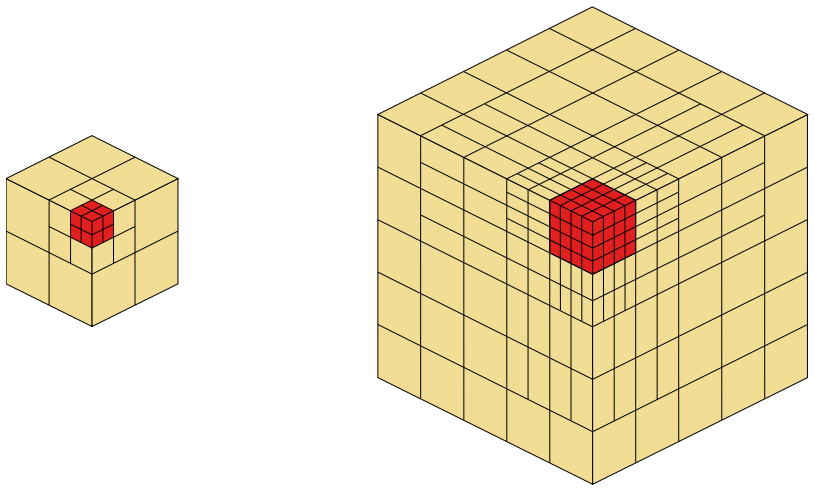


Fig. 3.21: Resulting refined mesh.

define the size of the finest elements as the refinement goal. Another initial model is given and the refinement instructions are given in order to obtain this element fineness without having overloading. These two meshes are given in Figure 3.22.

In Figure 3.22(a) the transition area consists in $L-1$ elements between a level 0 and a level L with nested refinements in one direction. For the same mesh fineness and with the full span scheme, there will be $\sum_{i=2}^{L} 2^{i-1}$ elements in Figure 3.22(b). As a reminder, the full span scheme requires to have mesh surfaces that are extended on the 2 elements that are adjacent to the refined area.

To summarize, there are more elements in the transition area, due to the non overloading constraint. This remark is of course valid for each of the three parametric directions. It is all the more the case since a refined element of level L comes from an initial element which has been cut in 2^L finer ones. From a level 2, each element refinement will create more than 4 finer elements. The mesh surfaces extension evoked just before will be larger than necessary. That said, the



(a) (b)

Fig. 3.22: Target refinement versus analysis-suitable refinement for quadratic meshes.

recommendations on the refinement smoothness in explicit go in the same direction. The more the refinement will be progressive, the better will be the mesh.

3.4.5 Critical time increment and computational cost aspect

The stable time increment is proportional to the size of the finest element. So if we refine a given mesh to obtain a finer one, the time increment will be decreased by 2 or more. The total computational cost will increase since the number of time steps is driven by the time increment.

The goal of local refinement is not to have even finer elements. It is rather to locally define coarser mesh areas in places where the strains or the contact are less important and which will not affect the overall quality of the solution. It will make it possible to start from a coarser model and to go up to the minimal size of elements that would have been necessary to apply in a global way to a non locally refined mesh. In conclusion, the time increment will not be smaller.

Computationnal cost of LR B-Spline basis function estimates The work of Cottrell et al. [28] was very often used to implement the IGA in implicit or explicit solvers. The shape functions and their derivatives calculation procedure has been designed to use the power and advantages of the global tensor product within the B-Spline or NURBS patch. The cost of calculating this type of shape function, although more complex than Lagrange polynomial functions, is not excessive. It

can be largely compensated by the gain in computing time provided by the critical time increment in explicit.

However, the local refinement methods implementation forced us to break the global tensor product. Local knot vectors per control point replaced the global knot vectors by patch. As a result, the estimation of shape functions and their derivatives can no longer be done by parametric direction but must be done in a unitary way. The initial calculation procedure must be replaced by the algorithm provided by Piegl and Tiller [86]. Since this procedure is much slower, it is still a limitation to the use of refined meshes in explicit codes.

This constraint is not only related to Locally Refined B-Splines. All refined Spline basis functions presented here can not be used with the classical shape function estimation method. Research efforts must be done to speed up this unitary estimate, by optimizing algorithms or by keeping in memory a certain number of invariant values.

3.4.6 Infinite plate with a hole

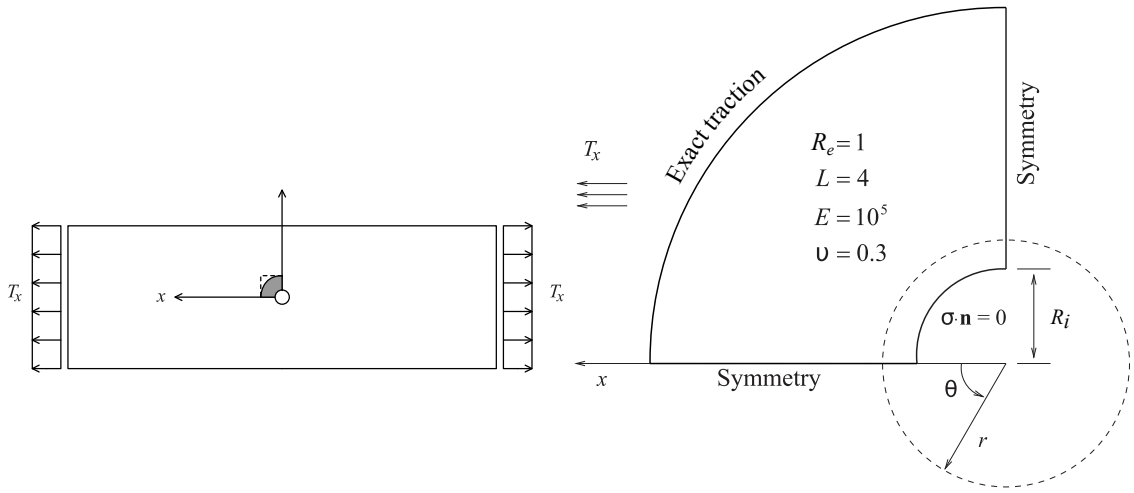
This study illustrates the phenomenon of stress concentration in an infinite plate with hole subject to an infinite traction. An infinite plate with a hole of diameter $d = 2\,\text{mm}$ at the center is subjected to a tensile stress $T_x = 10\,\text{MPa}$ assumed to be uniform and constant at its ends. The aim is to determine the stress concentration around the hole.

The problem is quasi-static linear in small deformation. The geometry and the boundary conditions being symmetrical, only a quarter is modeled. The infinite plate is thus represented by a finite 3D plate model in plane strains, see Figure 3.23. Choosing a circular outer edge instead of a square shaped edge allows to mesh the geometry with only one quadratic NURBS patch and to have C^1 continuity inside. It avoids the C^0 continuity between patches in the case of a square representation with two quadratic NURBS patches.

The analytic solution given by Timoshenko and Goodyear [100] provides the analytical displacement at the plate edge as follows:

$$U_x = \frac{T_x}{8\mu}\left(r(\kappa+1)cos(\theta) + \frac{2R^3}{r^2}((\kappa+1)cos(\theta)+cos(3\theta)) - 2\frac{R^4}{r^3}cos(3\theta)\right), \quad (3.7)$$

$$U_y = \frac{T_x}{8\mu}\left(r(\kappa-3)sin(\theta) + \frac{2R^3}{r^2}((1-\kappa)sin(\theta)+sin(3\theta)) - 2\frac{R^4}{r^3}sin(3\theta)\right), \quad (3.8)$$



(a) (b)

Fig. 3.23: Infinite plate with circular hole subjected to far-field tension and circular representation of the plate with boundary conditions, from [77]

with $\kappa = 3 - 4\nu$ for plane strains and $\mu = \frac{E}{2(1+\nu)}$. The analytical solution for stress fields is also given as follows:

$$\sigma_{xx} = T_x - T_x \frac{R^2}{r^2}(\frac{3}{2}cos(2\theta) + cos(4\theta)) + T_x \frac{3R^4}{2r^2}cos(4\theta), \tag{3.9}$$

$$\sigma_{yy} = -T_x \frac{R^2}{r^2}(\frac{1}{2}cos(2\theta) - cos(4\theta)) + T_x \frac{3R^4}{2r^2}cos(4\theta). \tag{3.10}$$

The displacements are computed at the boundary of the domain and applied as a Dirichlet boundary condition.

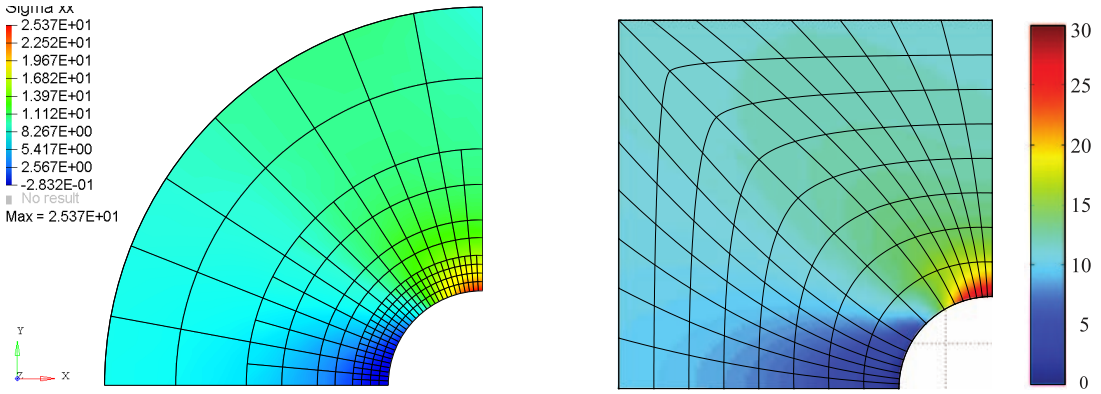
It can be verified that the maximum stress is at the edge of the hole on an axis perpendicular to the stress field direction, i.e. precisely at the position:

$$r = R_i = 10\,\text{mm}, \quad \theta = \frac{\pi}{2}. \tag{3.11}$$

The stress concentration coefficient is defined by the relation:

$$Kt = \frac{\sigma_{max}}{T_x} = \frac{30}{10} = 3. \tag{3.12}$$

The results for LR B-Spline mesh are given in Figure 3.24 and are compared to those obtained with a square representation and two NURBS patches, provided by Cottrell et al. [28].



(a) (b)

Fig. 3.24: Normal stress σ_{xx} for the infinite plate, a) for the quadratic LR B-Spline model and b) for quadratic NURBS model, from [28].

These simulation results are consistent with the analytical study and the numerical results given [28]. The maximum stress value is 25.37 MPa, giving a stress concentration coefficient of 2.5.

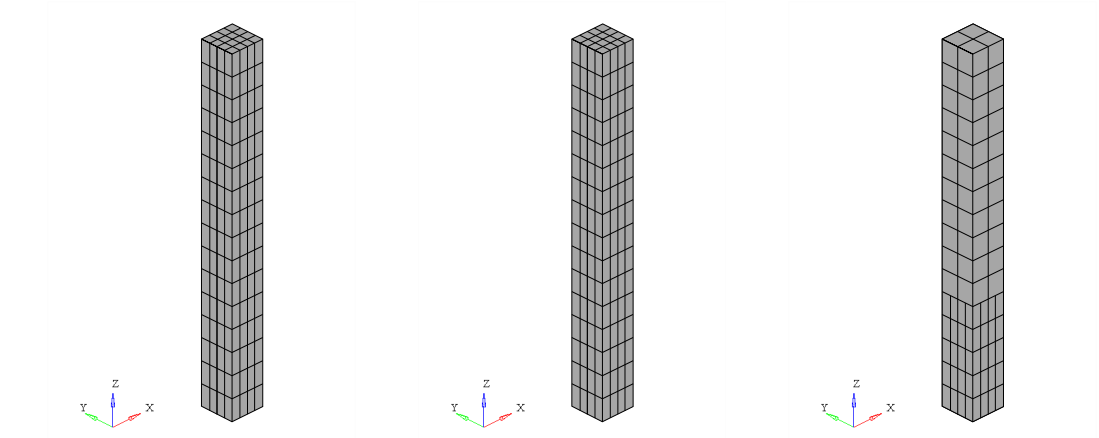
Applications - Numerical examples

Contents

In this chapter, few numerical examples will be reviewed, illustrating the B-Splines and LR B-Splines implementation capabilities.

4.1 Square Taylor bar impact

This simulation is the same as paragraph 2.3.2. Three models are used here and represent one quadrant of the rod, due to the inherent symmetry: C^1 quadratic B-Splines, C^1 quadratic LR B-Splines, both with only one patch, and one finite element model with 20 nodes quadratic FE (S20 in Radioss element library). Element densities are identical for the first and third models, that is 256 elements. The LR B-Spline model has the same element density in the bottom-part of the bar, but 50% coarser on the upper-part of the bar, which results in 124 elements in total. These meshes are given in Figure 4.1. The appropriate boundary conditions prescribed on each of the two symmetry planes for the problem are defined. On the bottom face of the bar, the vertical component of the kinematic fields is fixed. The other components are free. An initial vertical velocity of 227 $\mathrm{m.s^{-1}}$ is imposed on all the node/control points.

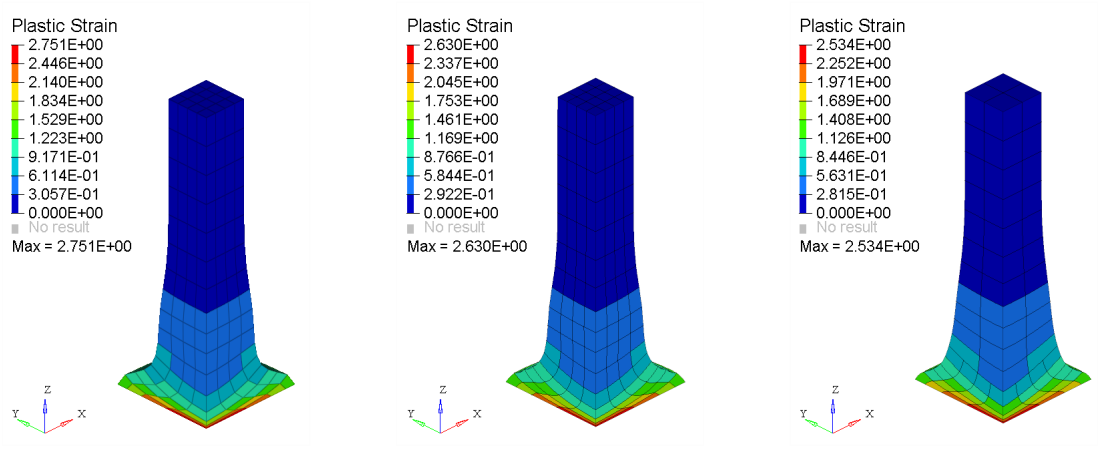


(a) (b) (c)

Fig. 4.1: Discretization used for the square Taylor bar case. a) FE 20-nodes quadratic model, b) C^1 quadratic B-Spline model and c) C^1 quadratic LR B-Spline model with local refinement.

Results and observations

Results are shown in Figure 4.2 and reproduce the mushrooming behavior observed in Rakvåg et al. [88]. From these figures it is clear that extremely high plastic strains



(a) (b) (c)

Fig. 4.2: Plastic strain on all three models at the end of the calculation. a) FE 20-nodes quadratic model, b) C^1 quadratic B-Spline model and c) C^1 quadratic LR B-Spline model with local refinement.

develop at the crushed extremity of the rod, close to the center of the bar, resulting in severe local mesh distortion. Kamoulakos [63] gives the total equivalent plastic strain expression at the center of the rod for the cylindrical rod case:

$$\varepsilon_p = \frac{C_E^2 - C_p^2}{C_E^2 C_p}(V - \frac{Y}{\rho C_E}) = 2.02, \tag{4.1}$$

with C_E and C_p respectively the elastic and plastic waves speeds and V the initial velocity of the rod. This value will be considered as a reference value in the square rod case.

	Quadratic FE S20	C^1 quad B-Spline	C^1 quad LR B-Spline
Number of elements	256	256	4×11+16×5=124
Number of nodes	1505	648	388
Relative time increment	1	6.25	6.25
Relative number of time steps	1	0.28	0.28
Relative total CPU time	1	0.56	0.66
Maximum plastic strain	2.751	2.630	2.534

Table 4.1: Results comparison for all three models. FE ones are used as a reference.

Table 4.1 groups informations about all meshes in terms of number of elements

and number of nodes. It also gives minimal time increment during the simulation, computational time and total number of time steps. For comparison purposes, we have considered as a reference the results obtained with the finite element model, values for other models are normalized with respect to these ones. This table shows that Spline based models are at least 40% faster. This is mainly due to a much higher time increment for those models, about 6 times higher, which greatly reduces the total number of time steps. The number of element is the same between the B-Spline model and the FE one, however the C^1 continuity inside the patch allows us to have fewer control points. The LR B-Spline model emphasizes this point even more because of the coarser discretization on the upper-part of the rod. Only the lower part is strongly deformed. The quality of the obtained solution is also better for the Spline models, with a maximum plastic strain closer to the reference value.

4.2 Axial crushing of a thin-walled beam

The dynamic axial crushing of a thin-walled rectangular box beam against a rigid wall is a typical benchmark in transient simulations. It is a conventional energy absorption device that is applied especially for frame structure of land and railway vehicles. Several experimental and numerical studies were done on this industrial case, see, e.g. Jusuf et al. [59]. and Figure 4.3(b). The purpose of this example is to study the discretization influence on simulation results, comparing a classical FE model and a B-Spline one, for geometric non-linearities with self-contact and buckling.

The model we consider is taken from Radioss example library. It represents an aluminium hollow beam, leaned on its lower face and crushed by a rigid wall on the upper face with a descending vertical initial velocity of 13.3 $\mathrm{m.s^{-1}}$. The boxbeam has a thickness of 0.914 mm, a hollow square section of 50.8 mm by 38.1 mm and a height of 203 mm, see Figure 4.3(a). A 0.1% random noise is introduced on the entire structure, on control points/nodes coordinates to trigger the buckling. This perturbation is different from the one used in Benson et al. [14], Belytschko et al. [10] where it is positioned at a specific height from the base and not on the entire box beam. As will be seen in the results, imposing such a perturbation triggers the buckling on the upper-part of the beam and not where the perturbation is imposed. Only a quarter of the beam is modeled due to the inherent symmetry of the problem. The appropriate symmetry boundary conditions are prescribed. The B-Spline parametrization of the quarter boxbeam is done with two patches, using one cubic element through the thickness and C^1 quadratic elements in the other directions. The FE model is composed of piecewise linear 8-nodes elements (HA8 in the Radioss element library), with a similar element density. A Type 7 interface is used to capture the self-contact during the crushing.

An isotropic elastic plastic material with user-defined isotropic hardening is used. Young's modulus is set to 60.4 GPa and the yield stress is 90 MPa. The initial density is 2.7×10^{-3} g.mm^{-3} and Poisson's ratio is 0.33. Boundary conditions and dimensions are given in Figure 4.3(a).

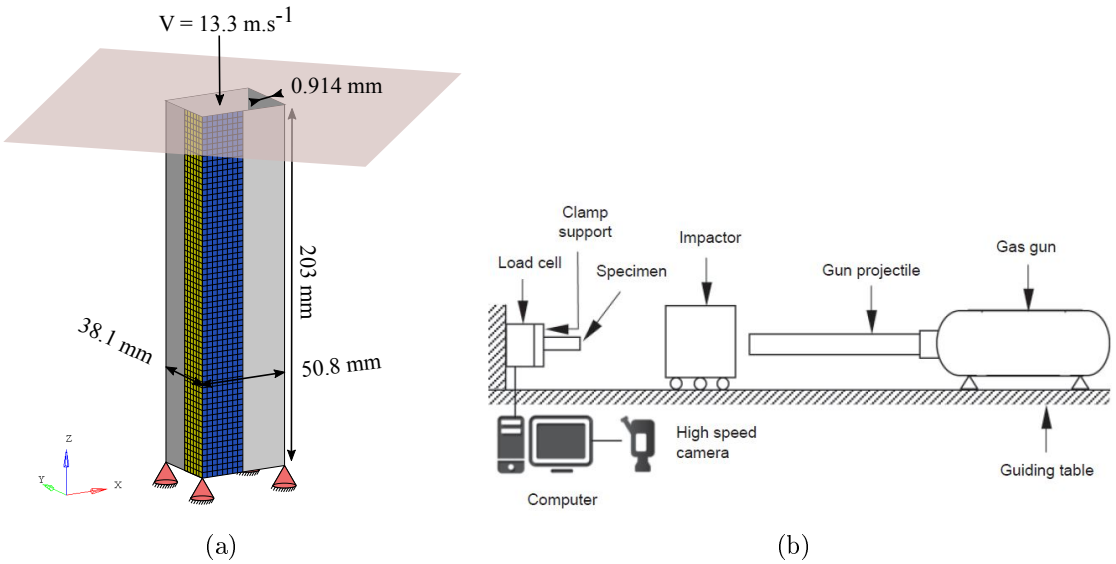


Fig. 4.3: a) Dimensions, boundary conditions and mesh for the B-Spline model. b) High speed impact testing machine, from Jusuf et al. [59].

Results and observations

The final deformed shapes are given in Figure 4.4. The computed models have been symmetrized for visualization purpose. We can see that both Radioss models have the same buckling mode: same number and same height of plies. The crushing response (resultant force) on the rigid wall is plotted in Figure 4.5. It shows at the very beginning of the computation a similar normal force intensity for both models. This peak corresponds to the first contact between the hollow beam and the rigid wall, which triggers the buckling.

4.3 Came-valve system

This example illustrates the same case as in paragraph 2.4.3. The cam and the valve are both discretized with 5 patches. For the LR B-Spline model, local

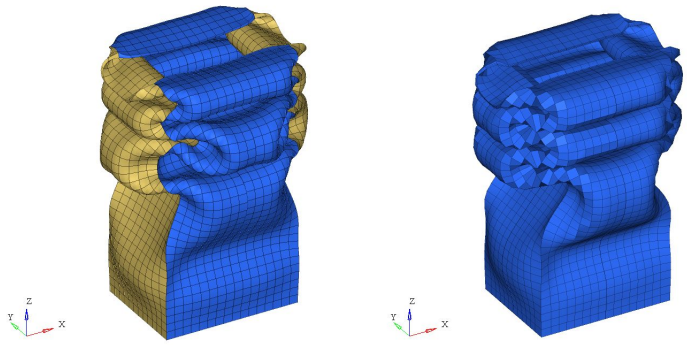


Fig. 4.4: Boxbeam shape deformations: deformed structure reconstruction using symmetry projection for B-Spline model and FE model.

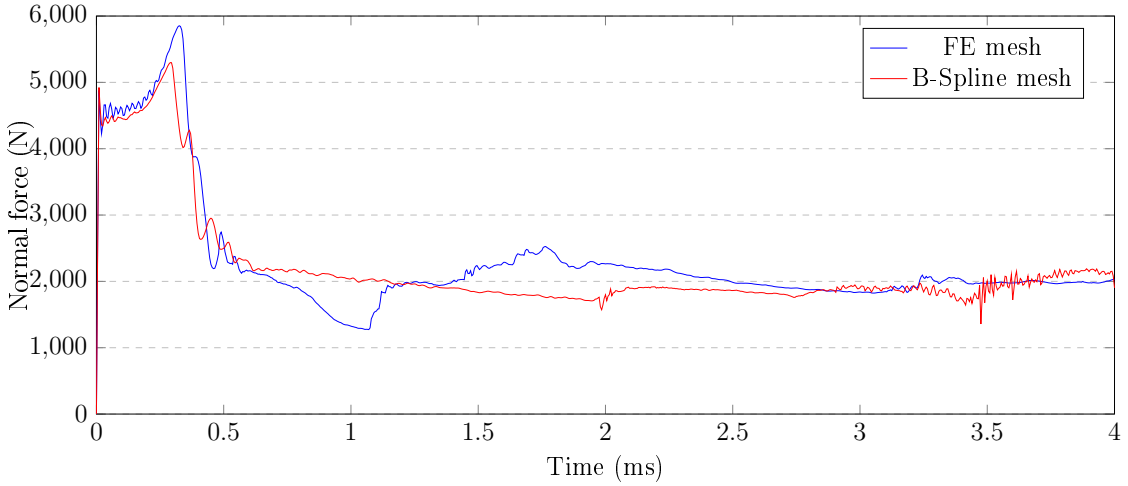


Fig. 4.5: Normal force on the impactor.

refinement has been performed on the outer row to better represent the contact surface with the valve. The FE model is meshed with piecewise linear bricks (HA8 in the Radioss element library).

Results and observations

Figure 4.8 provides the velocity of the valve's master node as a function of time. The raw results obtained are noisy for all models. This is due to the faceted description of the geometry in the contact algorithm, as explained in section 2.4.

The oscillations correspond to each application of a contact force to the valve. Note that for the LR B-Spline mesh, they have a smallest amplitude compared to the two others. The LR B-Spline element size on the circumference of the

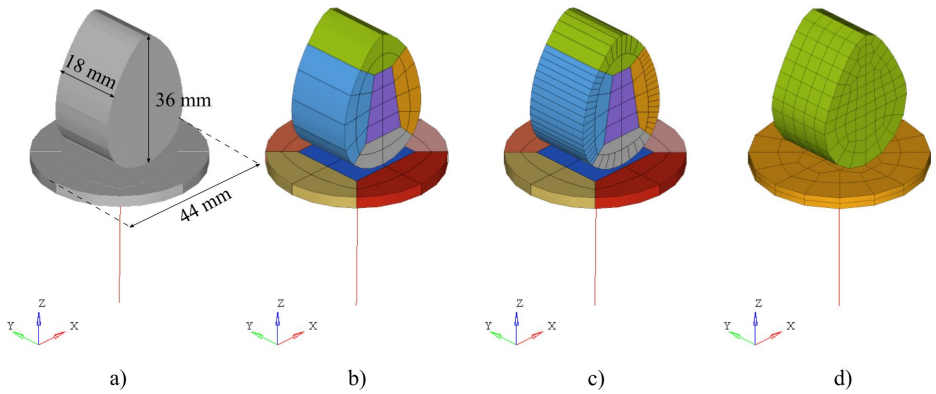


Fig. 4.6: Dimensions, boundary conditions and meshes. From left to right : a) Cam and valve's dimensions, b) C^1 quadratic B-Spline mesh, c) C^1 quadratic LR B-Spline mesh and d) FE linear 8-nodes mesh.

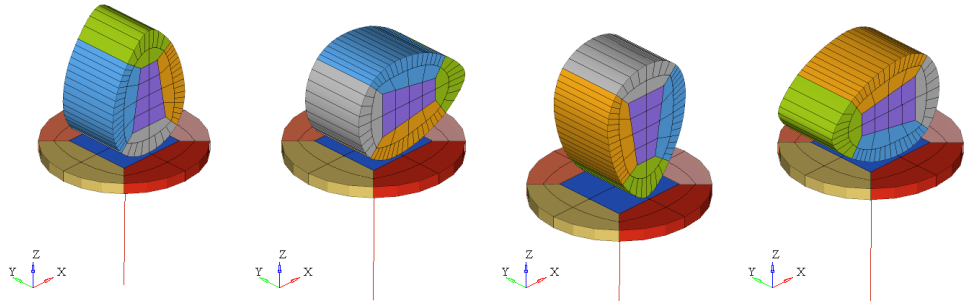


Fig. 4.7: Cam and valve positions during the simulation, for the LR B-Spline model.

cam is smaller than the FE one, consequently the LR B-Spline contact surface discretization is much finer. It results in a slightly smoother response. Due to the local refinement, the oscillations are drastically reduced.

A smooth velocity curve can be obtained by using a low pass filter. The filtering quality depends on the number of samples which in this case is the number of points computed by Radioss for each curve. The smooth velocity curves are shown on Figure 4.9. Results of all the models are nearly identical.

4.4 Stamping simulation

This example consists in the stamping process of a steel sheet. The stamping tools include a punch, a die and a blank holder. The initial thickness of the metal sheet

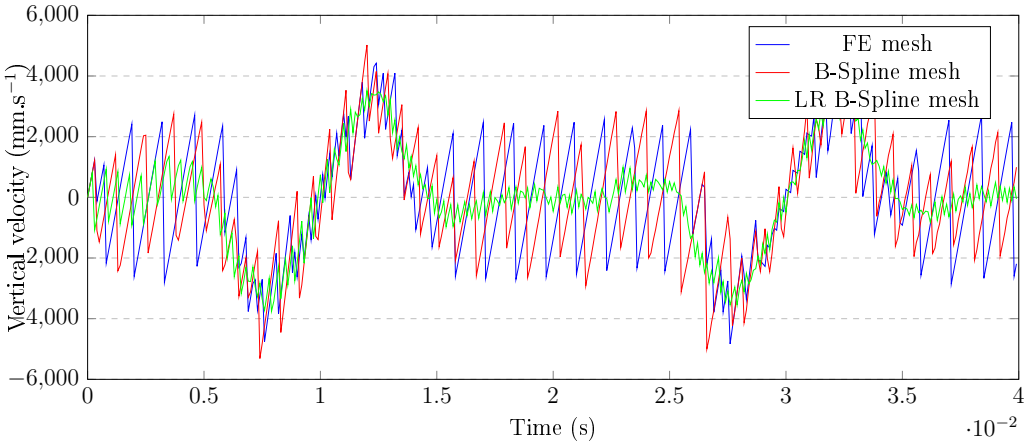


Fig. 4.8: Valve's master node vertical velocity.

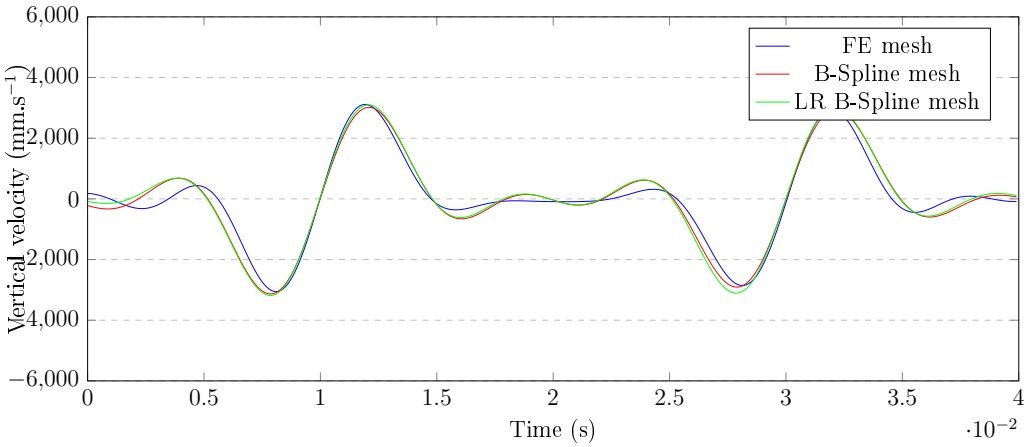


Fig. 4.9: Filtered valve's master node vertical velocity.

is 1 mm, the length is 483 mm and the width is 281.4 mm. A resultant force of 200 kN is vertically applied on the blank holder in order to flatten the sheet against the die. We apply an evolving vertical velocity on the punch with a maximum value of 5 m.s^{-1}. Taking symmetry into account, only one half of the structure is modeled. Parts, boundary conditions and loads are shown in Figure 4.10. The punch is shown in yellow, the blank holder in transparent green and the die in blue. The sheet is colored in orange.

An isotropic elastic plastic material with isotropic Jonhson-Cook hardening is used. The material parameters are 210 GPa for Young's modulus and 270 MPa for the yield stress. The initial density is 7.8×10^{-3} g.mm^{-3} and Poisson's ratio is 0.33. We consider four different models to represent the sheet: an adaptative piecewise linear 4-nodes shell model (QEPH in the Radioss element library), a piecewise linear 8-nodes model (HA8 in the Radioss element library), a C^1 quadratic B-Spline

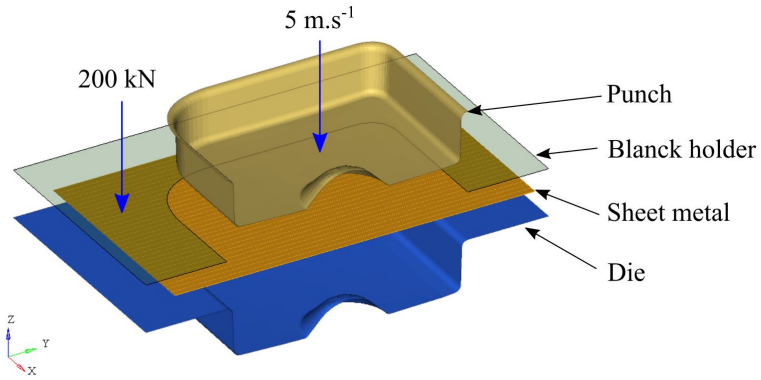


Fig. 4.10: Half-model exploded and boundary conditions.

model, and a C^1 quadratic LR B-Spline one. LR B-Spline model allows to reduce the number of elements from 2048 to 1326. The number of control points is reduced to 2574. All the three solid models use one element through the thickness. The other parts, the punch, the blank holder and the die are meshed with FE triangular rigid shell elements. Final shape deformations are given on Figure 4.11.

Results and observations

The final shape of the sheet is studied. Table 4.12(b) lists the amounts of draw-in in the X, Y and diagonal directions named by DX, DY1, DY2, DXY1 and DXY2 at punch travels of 69 mm, see Figure 4.12(a). In comparison with the two values obtained with FE models, B-Spline and LR B-Spline models seem to be a little stiffer than the FE shell one but a little softer than the FE solid one. Figure 4.13 shows that the most stressed zones are located at the edges of the punch. The Von Mises stress is up to 584.9 MPa for all four models, well above the initial elastic limit of the material. We can see in Figure 4.14 the maximum plastic strain for all four models, corresponding to the stressed areas.

4.5 Cylindrical tube under external pressure

This example illustrates the nonlinear buckling of an aluminum cylindrical tube under external pressure, with self-contact. Experimental tests have been performed by Farhat et al. [42], see Figure 4.15. Numerical simulations were performed with isogeometric shells by Benson et al. [14].

The cylindrical tube has an internal radius of 19 mm and an external radius of 19.7 mm, a total length of 177.8 mm. Two plugs at the ends of the tube apply an

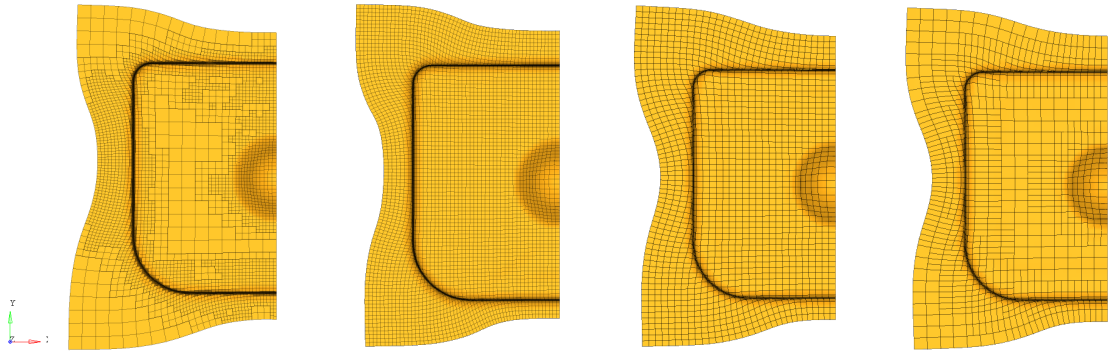
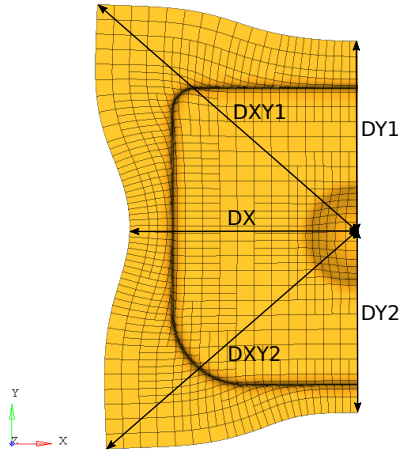


Fig. 4.11: Half-model shape deformation for all models, for a 69 mm punch travel. From left to right: the adaptative FE Shell model, the FE S20 model, the C^1 quadratic B-Spline model and the C^1 quadratic LR B-Spline model.



(a)

	DX	DY1	DY2	DXY1	DXY2
Adaptative FE Shell	238.0	192.8	187.4	348.3	357.5
FE S20	227.8	196.6	175.6	350.1	328.9
C^1 quad B-Spline	233.5	197.1	180.4	356.3	336.3
C^1 quad LR B-Spline	233.9	199.7	180.4	357.1	337.2

(b)

Fig. 4.12: Distances between corners and edges and the center of the sheet for a half-model shape deformation. b) Draw-in for the sheet (unit: mm) for a 69 mm punch travel. Results comparison for all four models.

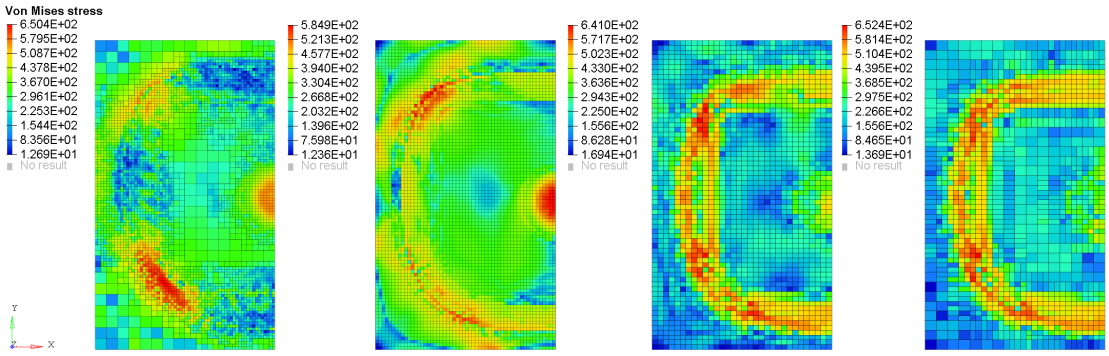


Fig. 4.13: Von Mises stress on all half-models, for a 69 mm punch travel. From left to right: the adaptative FE Shell model, the FE S20 solid model, the C^1 quadratic B-Spline model and the C^1 quadratic LR B-Spline model.

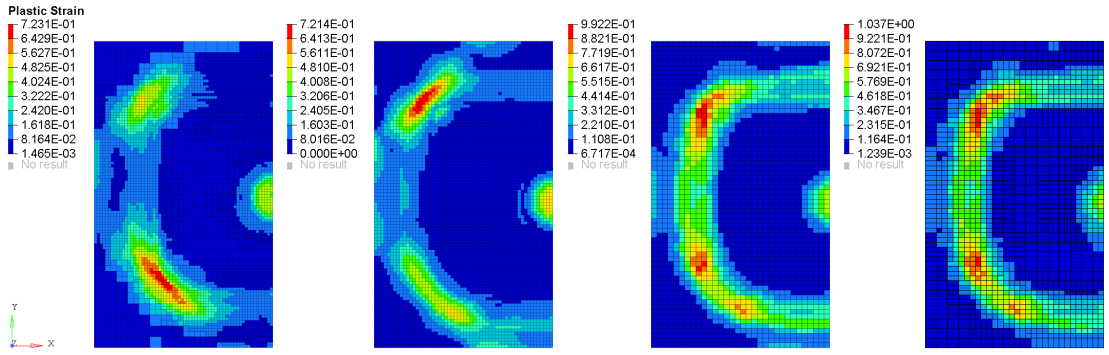


Fig. 4.14: Plastic strain on all half-models, for a 69 mm punch travel. From left to right: the adaptative FE Shell model, the FE S20 model, the C^1 quadratic B-Spline model and the C^1 quadratic LR B-Spline model.

axial compression, which is correlated with the imposed pressure, see Figure 4.15. These plugs are inserted over a length of 25.4 mm. An external pressure is applied to the cylinder, reproducing the pressure difference between the inside and the outside. This pressure is carried out in a ramp of 1 ms then is held constant until the equilibrium of the structure.

An isotropic elastic plastic material with a user-defined isotropic hardening is used. For this simulation, the alloy has a Young's modulus of 69.5 GPa, a yield stress of 227 MPa. The initial density is 1073 kg.m^{-3} and Poisson's ratio is 0.3.

Figure 4.16(a) illustrates the boundary conditions and the dimensions. Only one half is modeled due to the symmetry. The compressive loading imposed by the fluid pressure on the end plug is modeled by a distributed force on the planar end surface. The kinematic boundary conditions corresponding to the rigid contact between the tube and the plug are modeled by imposing zero radial displacement on

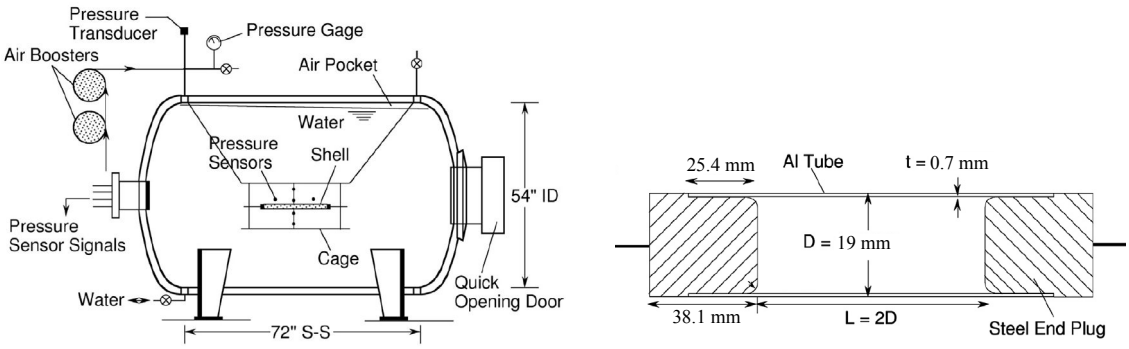


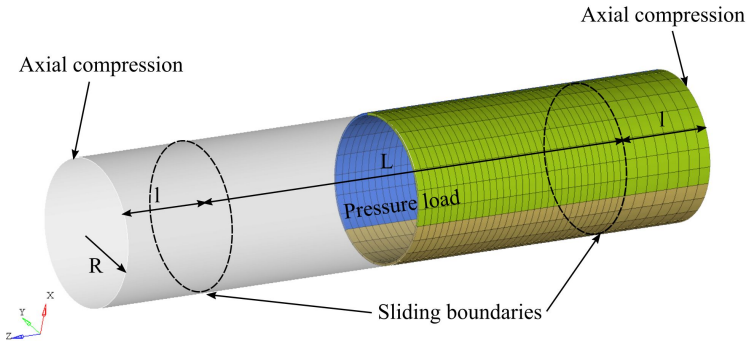
Fig. 4.15: Kyriakides' experimental protocol and boundary conditions. Pictures taken from Farhat et al. [42]. Note that dimensions on the right picture were converted to the metric system for consistency.

the corresponding surface. The cylinder is parametrized using three C^1 quadratic NURBS patches, with C^0 interfaces between each of them. A C^0 circumferential line is added to represent the boundary between the constrained and free portions of the tube. The imposed external pressure is applied with a ramp of 1 ms then is held constant until the equilibrium of the structure is reached.

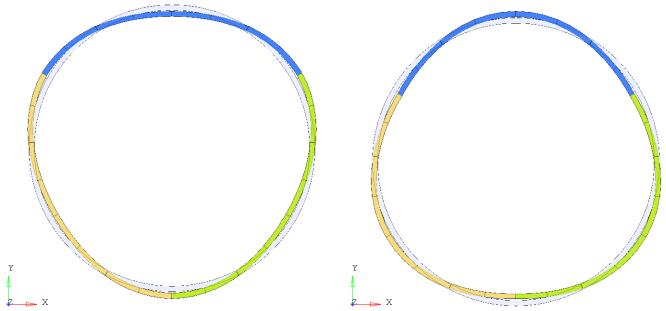
We use a mesh composed of 960 quadratic elements, keeping the fact that only one element is used through the tube's thickness. Two kinds of geometric perturbation are introduced, distributed on the cylinder section in order to trigger the buckling in the third mode. These distributed noises are defined scaling the x and y coordinates of the control points by $1 + \frac{1}{100}cos(3\theta)$ and $1 + \frac{1}{100}cos(3 * (\theta - \frac{\pi}{3}))$. The first noise configuration will push outwards the C^0 patch interfaces, pushing inwards the middle of each patch. The second configuration does the opposite. A Type 7 interface is used to capture the self contact of the inner surface of the cylinder. Cross sections of the two perturbed geometries are shown on Figure 4.16(b).

Results and observations

Simulation results and final deformed configurations are presented in Figure 4.18 and show relatively good agreement. The experimental results showed a critical pressure of 2.827 MPa. This experimental pressure is used as a reference. For the first perturbation, the critical pressure needed to trigger the buckling on the third mode is 2.930 MPa, i.e. a corresponding error of 3.6% with the analytic reference. The final shape shows the apparitions of three kinks at the lobes, certainly due to the C^0 continuity accross patch interfaces. The second perturbation leads to a different final shape. The kinks are still located at the patch interfaces. The deformed shape is closer to the experimental one in this case, since the experimental deformed configuration has a kink at the triple contact point on the cylinder axis, see Figure 4.18.



(a)



(b)

Fig. 4.16: Boundary conditions, mesh and noise repartitions (highlighted at 5% for better visualization). The original shape is also given.

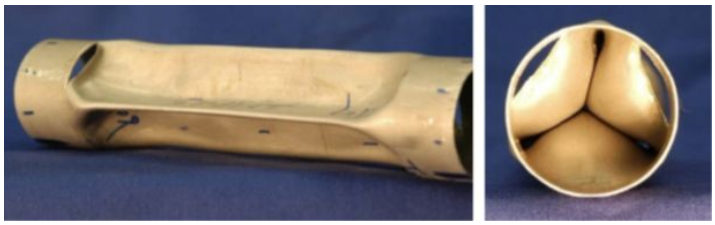
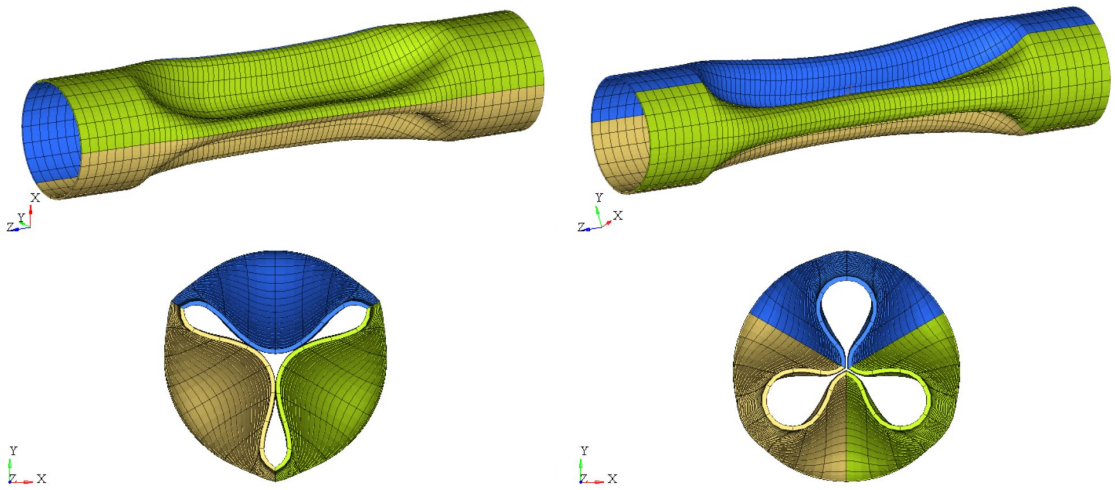


Fig. 4.17: Kyriakides' experimental result, from [14].

The buckling pressure with the second configuration is relatively higher. We obtain a value of 3.448 MPa, which corresponds to an error of 22%. This difference in the buckling pressure could be an indication that the mesh is not fine enough to obtain a converged solution. This situation might be amplified by the relative alignment of the geometric perturbation and the C^0 patch interfaces.

A mesh twice finer in the circumferential direction has been used with both perturbations and show identical results. Both simulation results show the same final deformed shape, which is different from the previous ones, but much closer



(a) (b)

Fig. 4.18: Buckling of the meshes with the two noise configuration. a) Kinks appear on the lobes with a buckling pressure of 2.930 MPa. b) Kinks appear at the cylinder's center with a buckling pressure 2.827 MPa. Both figures are extracted when the equilibrium of the structure is reached.

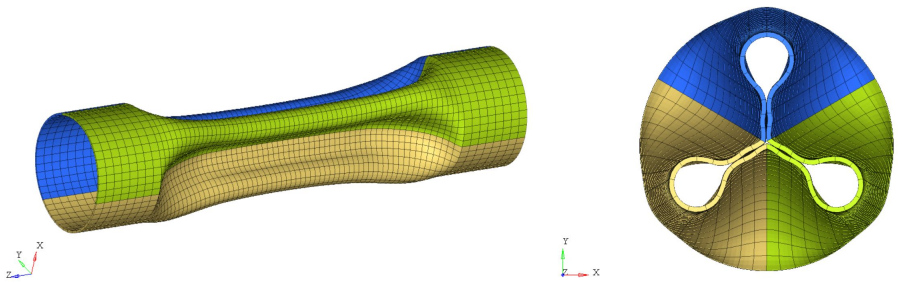


Fig. 4.19: Same deformed shape for both finer meshes. The appearance of sidelobes buckling can be noticed. Buckling pressure of 2.41 MPa, when the equilibrium of the structure is reached.

to the experimental one. We can note that with such discretization the geometric perturbation is not mandatory anymore to trigger the buckling in the third mode. The critical pressure required to cause the buckling is lowered to 2.41 MPa, which coresponds to an an error of 14.7%. This is lower than the experimental buckling pressure. Nevertheless, we can remark that the cylinder bounces back outward after its implosion, and creates the secondary lobes near the axis, see on Figure 4.19. These secondary lobes are missing from the experimental results, and have been reported by Benson et al. [14].

4.6 Cardboard box drop test

This example models the drop test of a simplified cardboard box on its corner and edge. The interest of this example is that it combines in the same model several contact interfaces and a hybrid discretization with FE elements and B-Splines. The cardboard is composed of several parts with different materials. The box inners, colored in pink on Figure 4.20, are represented by aluminium blocks and are connected at their sides by rubber pads (in blue) to a steel housing (in light red). This housing rests at its corners on styrofoam cushions (in green). Finally, a cardboard (in orange) packs all the parts. The purpose of this example is to check that the packing of the inners is efficient: energy absorption...

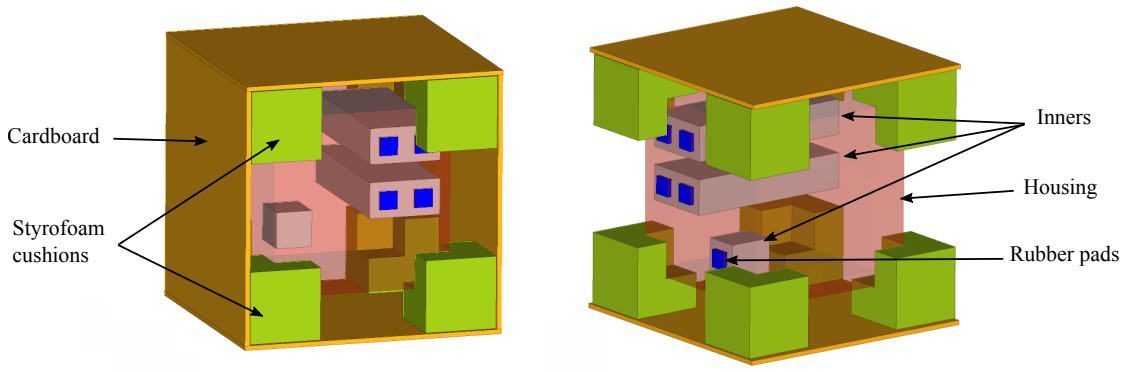


Fig. 4.20: The cardboard box and its internal parts. Some parts of the cardboard were removed for a better visualization.

The cardboard box and the styrofoam cushions are discretized with C^1 quadratic B-Splines. The housing is discretized with piecewize linear shell elements. For all the other parts, piecewize linear FE bricks are used. Type 7 contact interfaces are defined between the cardboard box, the styrofoam cushions and the housing. All other part connections (e.g. between inners and rubber pads) are imposed by merging nodes with compatible meshes. An initial vertical velocity of 5 $\mathrm{m.s^{-1}}$ is set as well as a gravity field, which represents the drop. A rigid wall is fixed in front of the corner or the edge of the cardboard, depending on the case considered.

Material models used in this example are elastic for inners, elastic perfectly plastic for the cardboard and for the housing, foam-plastic for all the styrofoams, and hyperelastic for the rubber. The cardboard is considered as isotropic for simplicity.

Results and observations

Simulation results for the corner and the egde drop tests are given on Figure 4.21 and Figure 4.22. As said previously, the analysis of those results is focused on the energy balance of the inner parts. Energy balances for each test can be plotted to compare the total kinetic energy, the total internal energy, the contact energy and the inners internal energy, in a logarithmic scale. Results show a conversion between kinetic and internal energy during the drop test. Most of the energy is absorbed by the styrofoams and the cardboard and let the housing and the inners safe. Left figures show the internal force magnitude on the inners. We observe that the highest value is at least six times lower than that of the other parts of the model.

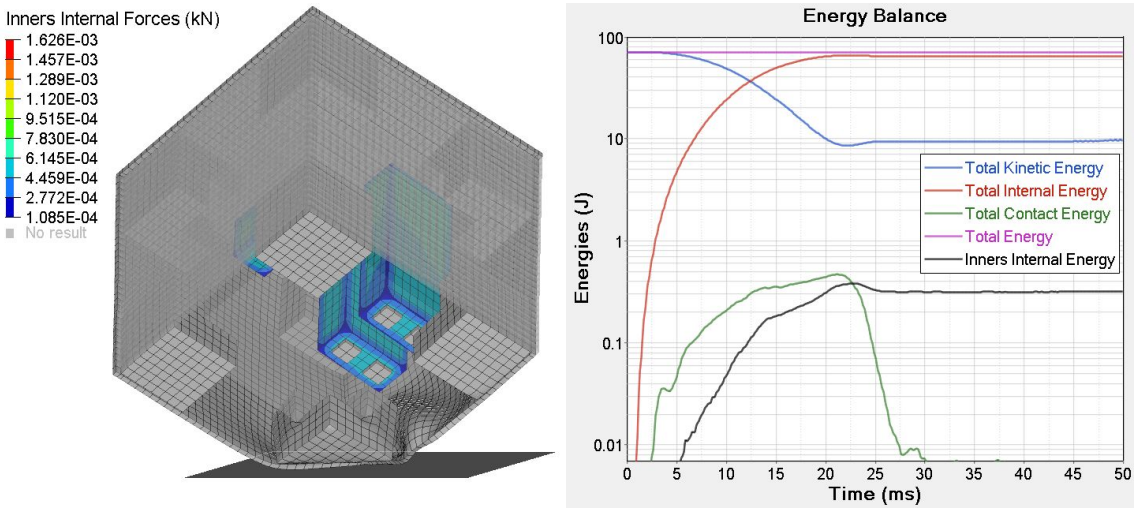


Fig. 4.21: Deformation and energy balance for a corner drop test.

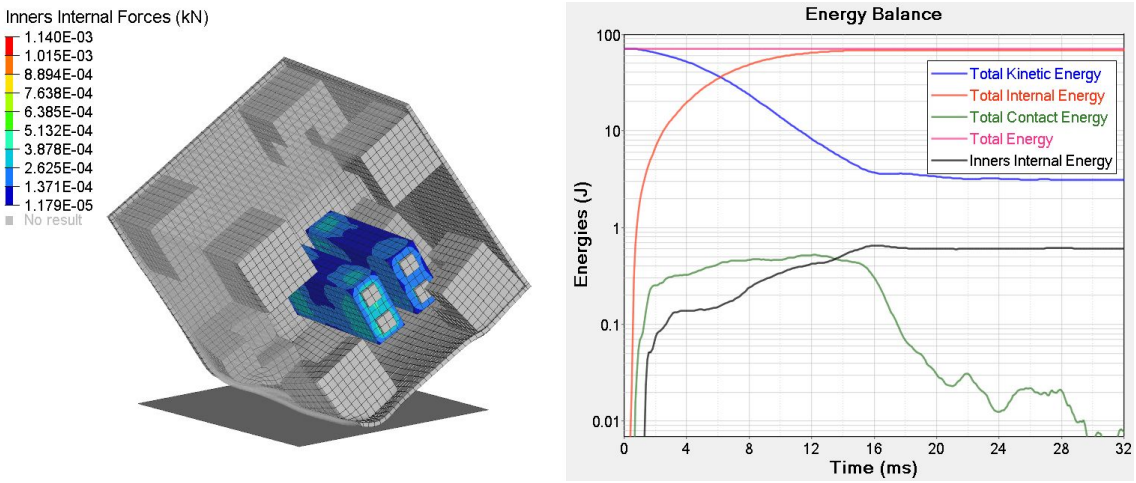


Fig. 4.22: Deformation and energy balance for an edge drop test.

Conclusion

This thesis was devoted to the integration of isogeometric analysis in an existing industrial explicit finite element software. All the needed ingredients for a complete simulation of dynamic phenoma such as stamping or crash have been identified, studied and modified according to the specificities of the IGA.

We were able to define heuristic estimates for the critical time increment associated to these new elements. These estimates are initially given for conventional finite elements and have been adapted to suit our basis functions. Simulations have shown that the time increment of simulations obtained with isogeometric elements increases significantly. This gain is a very strong argument to put forward IGA in solvers, especially explicit ones.

An important part this work was dedicated to the modification of the contact interface. We have been able to integrate IGA, only addressing the representation and discretization of Spline surfaces. Spline surface discretization was simply done by a set of linear facets. Nevertheless, it allows a better representation of surfaces compared to what is obtained with conventional finite elements with similar initial element densities. We showed the benefits given by this simple contact formulation in some industrial cases.

We highlighted and analyzed several Spline basis functions allowing local refinement with their characteristics, taking into account the implementation constraints of each of them. We implemented the LR B-Spline basis functions to define locally refined tridimensional models. We showed that the previously given refinement schemes were not sufficient to guarantee a set of non-overloaded elements, especially for quadratic cases. The refinement complexity that can be found in industrial models is such that it is beyond the scope of the study established until then. We introduced an improved full span scheme, based on the full span scheme. More complex refinement shapes such as L-shaped refinement, or more general shapes with an inward corner, have been taken into account. The improved full span scheme thus appears to be the appropriate method for dealing with refined models. It appeared in the LR B-Splines implementation that it was indeed possible to have a native integration within Radioss keeping the engineers usage.

The IGA in Radioss has improved the quality of the results obtained compared to traditional FEA. B-Splines also provide better accuracy and will quickly give a solution without compromising the quality of the result. However, the LR B-Splines computational costs are for the moment not satisfying but improve the quality of solution.

A fair comparison of IGA and FEA is difficult when having FEA engineers habits. We can state without compromises that IGA can show better accuracy or better time step size and on some cases both at the same time. Improving its computational cost to reach industrial solvers standard beyond the work shown in this thesis will make it a very good competitive method compared to FEA.

Further work

This implementation is a first step towards the IGA integration, which can be continued by a work related to the performance of the algorithms and technologies used. Computational costs have to be analyzed and treated as a target to make IGA an industrial solution. Several tracks have already been identified, for example by storing in memory some invariant values rather than recalculating them a large number of times. The cost of occupying memory is to be quantified but the gain in computational time could be important. Another way could be by considering the use of Bézier extraction instead of having a native IGA integration. This method also eliminates a large number of calculations since some values are invariant and can be stored in memory.

We could denote that, as for conventional finite elements, boundary elements are the most restrictive ones for the time increment due to the smaller size of the basis function support. During the different simulations, we also noticed that the safety factor commonly used in FEA is too conservative for IGA. Other methods, specific to IGA, could be studied to improve even more the time increment to make it even more competitive.

Regarding contact, we could implement accurate algorithms to be able to use the real Spline surface. This would bring more robust solutions of higher quality. The efforts to provide are not only at the geometric level but also on the contact formulation which must be effective enough to be a potential competitor to the existing ones.

As a continuation of this work, we can introduce an isogeometric shell formulation with a high degree and regularity. Indeed, to simulate a car crash test, more than 90% of the geometry is meshed with shell elements. Finally, new reduced integration methods can be developped, which will avoid hourglass modes, while

providing a large gain in computational time, particularly for higher degrees.

Appendix A

\IGE3D and \PROP\TYPE47 (IGA3D) cards

Block Format Keyword

/IGE3D – Isogeometric volumic elements

Description

Defines an isogeometric element with an arbitrary number of control points which is possibly refined.

Format

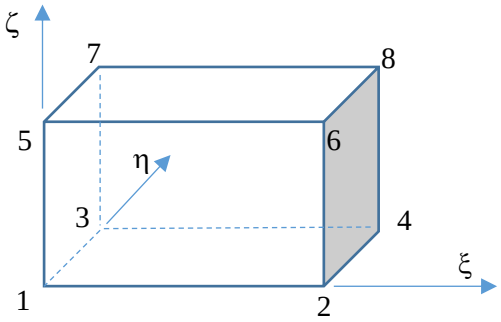
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
/IGE3D/part_ID									
Ige3D_ID	i1	i2	i3	NNode	Rafdir1	Rafdir2	Rafdir3		
Node_IDijk	…	…							

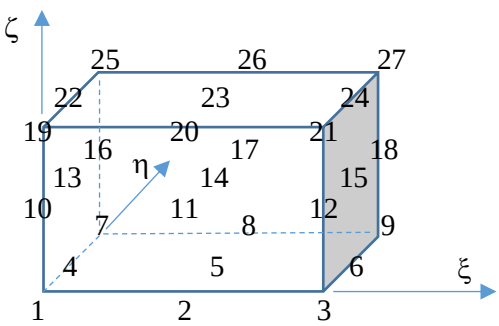
– Flag Definition

Field	Contents
part_ID	Part identifier of the block (Integer, maximum 10 digits)
ige3D_ID	Element identifier (Integer)
i1	Index of 1st knot in the knot vector corresponding to the element
i2	Index of 2nd knot in the knot vector corresponding to the element
i3	Index of 3rd knot in the knot vector corresponding to the element
NNode	Number of control points of the element
Rafdir1	Number of refinement in the 1st direction
Rafdir2	Number of refinement in the 2nd direction
Rafdir3	Number of refinement in the 3rd direction
Node_IDijk	Control points Id

– Comments

- The knot vector for the patch is given in the property set attached to the part
- The control points order is described in the figure below, for linear and quadratic elements :





Block Format Keyword

/PROP/TYPE47 – 3D Isogeometric property Set

Description

This property set is used to define the general solid property set.

Format

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
/PROP/TYPE47/prop_ID/unit_ID or /PROP/IGA3D/prop_ID/unit_ID									
prop_title									
IntRule	RafRule								
d1	d2	d3	NCP1	NCP2	NCP3				
$knot^1 1$		$Knot^1 2$		$Knot^1 3$		…		…	
…		…		…		…			
$Knot^2 1$		$Knot^2 2$		$Knot^2 3$		…		…	
…		…		…		…			
$Knot^3 1$		$Knot^3 2$		$Knot^3 3$		…		…	
…		…		…		…			

Flag Definition

Field	Contents	SI Unit Example
prop_ID	Property identifier (Integer, maximum 10 digits)	
unit_ID	Optional unit identifier (Integer, maximum 10 digits)	
prop_title	Property title (Character, maximum 100 characters)	
IntRule	Integration Rule (Integer) = 0: default, set to 2 = 1: reduced = 2: full	
RafRule	Refinement Rule (Integer) = 0: default, set to 1 = 1: LR-NURBS = 2: Truncated Hierarchical NURBS	
d1	Polynomial degree in 1^{st} direction (Integer)	

d2	Polynomial degree in 2^{nd} direction (Integer)	
d3	Polynomial degree in 3^{rd} direction (Integer)	
NCP1	Total number of control points in the 1^{st} direction of the patch	
NCP2	Total number of control points in the 2^{nd} direction of the patch	
NCP3	Total number of control points in the 3rd direction of the patch	
$Knot^{j}_{i}$	i^{th} parameter in direction j, j=1..3	

Comments

knot vector dimension is equal to N = NCP + d + 1, where NCP is the number of control points in the given direction and d is polynomial degree in this direction.

Appendix B

Rhino and the RhinoNURBS tool package

Rhino is a powerful and versatile 3D modeler which is able to generate B-Spline and NURBS geometries and in which it is possible to implement new procedures. An additionnal plugin was created to export Abaqus data files in the IGA context, see [3].

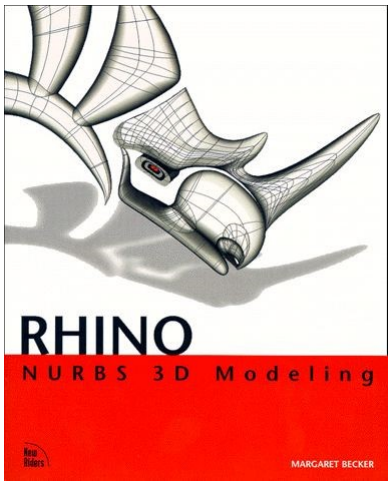


Fig. B.1: RhinoNURBS logo.

This work has been used to include an additional data file output option, in the Radioss format (0.rad and 1.rad files). The existing Visual Basic project has been modified, including the output file write format, and the ExportFile options have been changed to leave only the Radioss part, as can be seen in Figure B.2.

The RhinoNURBS plugin

Rhino's features for creating 3D isogeometric geometries are quickly described in this section for a simple case. Specific features such as the control point insertion

or the increase of polynomial degrees will not be described here. All the necessary information can be found in [3]. The reader will also be able to find the information necessary for the 2D isogeometric model generation, which is not used in this work.

3D B-Splines or NURBS geometry generation

The additionnal function uses the "Radioss-FileExport_Command.vb" command and the "RadiossFileExport_Class.vb" class. This command requires the selection of a NURBS surface or polysurface as input and returns all the NURBS parameters needed to create the 0.rad and 1.rad files for 3D NURBS geometry.

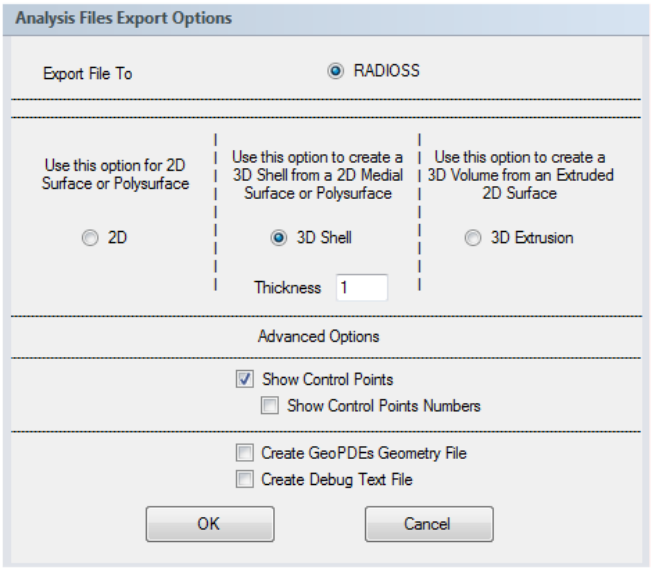


Fig. B.2: 0.rad dataset export window.

3D geometries can be obtained by extruding a surface along a straight or curved line, or by a "double extrusion" with a thickness parameter h, see Figure B.3.

Once the "RadiossFileExport_Command.vb" command has been successfully executed, the "Radioss Files Export" module uses the "RadiossFileExport_GUI.vb" GUI.

Boundary conditions defintion

The graphical interface "RadiossFileExport_GUI.vb", see Figure B.4, is used to indicate the displacement boundary conditions.

Other options such as large deformations (NLGEOM) or contact options are present on the window but are not used for Radioss. The same is true for loading

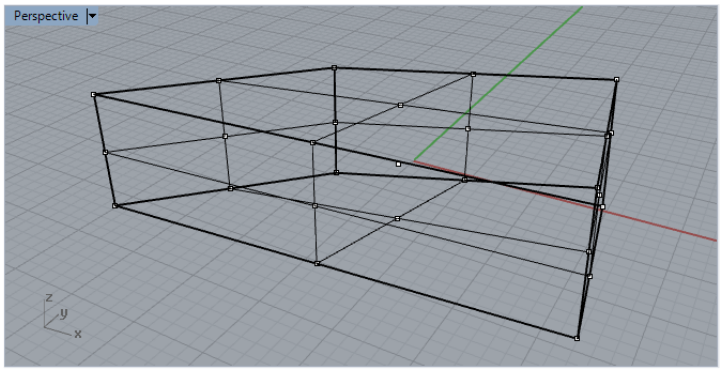


Fig. B.3: 3D geometry generated by double extrusion of a 2D NURBS surface.

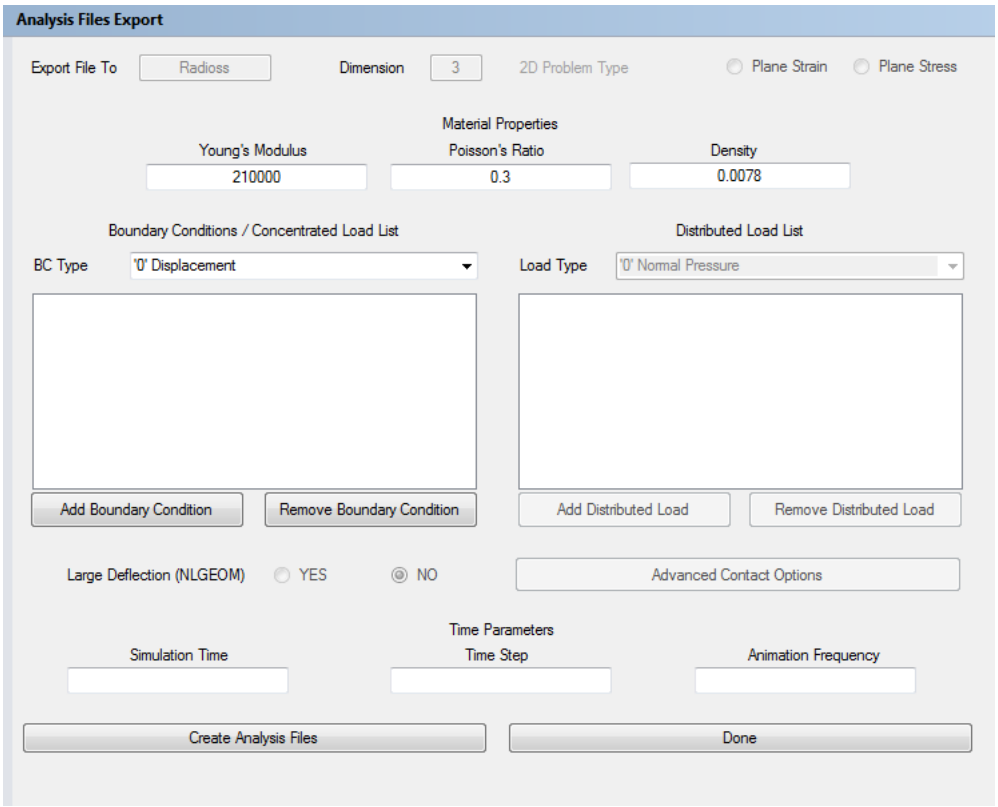


Fig. B.4: Rhino dataset export window.

boundary conditions.

The "Add Boundary Condition" button uses the "GetSelectedFreeEdgeIndex_Command.vb" command and the "BoundaryCondition_GUI.vb" GUI.

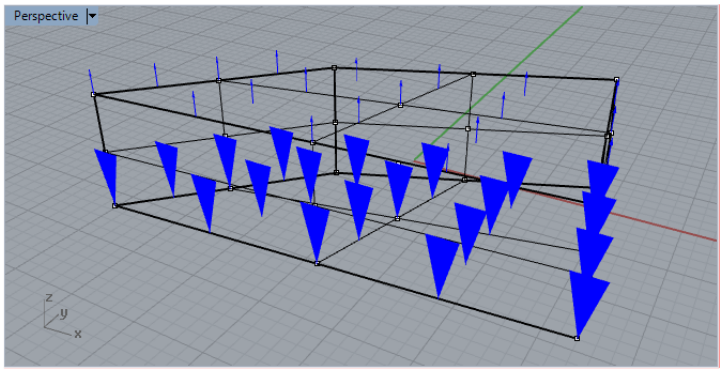


Fig. B.5: Illustration of boundary condition applied in Rhino.

Time parameters of the simulation

The graphical interface "RadiossFileExport_GUI.vb" also includes some time parameters. In this part is given the simulation time and the file export frequency. It is also possible to give a specific time increment value, but this option is not used anymore and will be removed.

Once all the geometric, material and temporal parameters have been filled, the class "RadiossFileExportRAD.vb" is called by the "Create Abaqus Files" button. The 0.rad and 1.rad files are then generated.

Bibliography

[1] C Adam, S Bouabdallah, M Zarroug, and H Maitournam. Stable time step estimates for NURBS-based explicit dynamics. *Computer Methods in Applied Mechanics and Engineering*, 295:581–605, 2015.

[2] C Adam, TJR Hughes, S Bouabdallah, M Zarroug, and H Maitournam. Selective and reduced numerical integrations for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:732–761, 2015.

[3] H Al Akhras. Développement d'éléments finis isogéométriques NURBS dans le code de calcul Abaqus. 2012.

[4] H Al Akhras, T Elguedj, A Gravouil, and M Rochette. Génération de modèles NURBS volumiques issus de la CAO pour l'Analyse Isogéométrique. In *12e Colloque national en calcul des structures*, 2015.

[5] H Al Akhras, T Elguedj, A Gravouil, and M Rochette. Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models. *Computer Methods in Applied Mechanics and Engineering*, 307:256–274, 2016.

[6] H Al Akhras, T Elguedj, A Gravouil, and M Rochette. Towards an automatic isogeometric analysis suitable trivariate models generation-Application to geometric parametric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:623–645, 2017.

[7] Y Bazilevs, C Michler, VM Calo, and TJR Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196(49):4853–4862, 2007.

[8] Y Bazilevs, VM Calo, JA Cottrell, JA Evans, TJR Hughes, S Lipton, MA Scott, and TW Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5):229–263, 2010.

[9] FB Belgacem, P Hild, and P Laborde. The mortar finite element method for contact problems. *Mathematical and Computer Modelling*, 28(4):263–272, 1998.

[10] T Belytschko, WK Liu, B Moran, and K Elkhodary. *Nonlinear finite elements for continua and structures*. John wiley & sons, 2013.

[11] DJ Benson. Stable time step estimation for multi-material Eulerian hydrocodes. *Computer methods in applied mechanics and engineering*, 167(1-2): 191–205, 1998.

[12] DJ Benson and JO Hallquist. A single surface contact algorithm for the post-buckling analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering*, 78(2):141–163, 1990.

[13] DJ Benson, Y Bazilevs, E De Luycker, MC Hsu, M Scott, TJR Hughes, and T Belytschko. A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM. *International Journal for Numerical Methods in Engineering*, 83(6):765–785, 2010.

[14] DJ Benson, Y Bazilevs, MC Hsu, and TJR Hughes. Isogeometric shell analysis: the Reissner-Mindlin shell. *Computer Methods in Applied Mechanics and Engineering*, 199(5):276–289, 2010.

[15] DJ Benson, Y Bazilevs, MC Hsu, and TJR Hughes. A large deformation, rotation-free, isogeometric shell. *Computer Methods in Applied Mechanics and Engineering*, 200(13):1367–1378, 2011.

[16] DJ Benson, S Hartmann, Y Bazilevs, MC Hsu, and TJR Hughes. Blended isogeometric shells. *Computer Methods in Applied Mechanics and Engineering*, 255:133–146, 2013.

[17] DJ Benson, D Bhalsod, S Hartmann, P Ho, L Li, W Li, A Nagy, and I Yeh. Isogeometric Analysis in LS-DYNA: Using CAD-Geometry for Numerical Simulation. In *10th European LS-DYNA Conference. Würzburg, Germany, June*, 2015.

[18] MJ Borden, MA Scott, JA Evans, and TJR Hughes. Isogeometric finite element data structures based on Bézier extraction of NURBS. *International Journal for Numerical Methods in Engineering*, 87(1-5):15–47, 2011.

[19] R Bouclier, T Elguedj, and A Combescure. Locking free isogeometric formulations of curved thick beams. *Computer Methods in Applied Mechanics and Engineering*, 245:144–162, 2012.

[20] R Bouclier, T Elguedj, and A Combescure. Efficient isogeometric NURBS-based solid-shell elements: Mixed formulation and B-method. *Computer Methods in Applied Mechanics and Engineering*, 267:86–110, 2013. ISSN 00457825.

[21] R Bouclier, T Elguedj, and A Combescure. On the development of nurbs-based isogeometric solid shell elements: 2d problems and preliminary extension to 3d. *Computational Mechanics*, 52(5):1085–1112, 2013.

[22] R Bouclier, T Elguedj, and A Combescure. Development of a mixed displacement-stress formulation for the analysis of elastoplastic structures under small strains: Application to a locking-free, nurbs-based solid-shell element. *Computer Methods in Applied Mechanics and Engineering*, 295:543–561, 2015.

[23] A Bressan. Some properties of LR-splines. *Computer Aided Geometric Design*, 30(8):778–794, 2013.

[24] A Bressan and B Jüttler. A hierarchical construction of LR meshes in 2D. *Computer Aided Geometric Design*, 37:9–24, 2015.

[25] G Cocchetti, M Pagani, and U Perego. Selective mass scaling and critical time-step estimate for explicit dynamics analyses with solid-shell elements. *Computers & Structures*, 127:39–52, 2013.

[26] N Collier, L Dalcín, and VM Calo. PetIGA: High-performance isogeometric analysis. *arXiv preprint arXiv:1305.4452*, 2013.

[27] JA Cottrell, TJR Hughes, and A Reali. Studies of refinement and continuity in isogeometric structural analysis. *Computer methods in applied mechanics and engineering*, 196(41-44):4160–4183, 2007.

[28] JA Cottrell, TJR Hughes, and Y Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley Publishing, 1st edition, 2009. ISBN 0470748737, 9780470748732.

[29] C de Falco, A Reali, and R Vázquez. GeoPDEs: a research tool for Isogeometric Analysis of PDEs. *Advances in Engineering Software*, 42(12):1020–1034, 2011.

[30] L De Lorenzis, I Temizer, P Wriggers, and G Zavarise. A large deformation frictional contact formulation using NURBS-based isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 87(13):1278–1300, 2011.

[31] L De Lorenzis, P Wriggers, and G Zavarise. A mortar formulation for 3D large deformation contact using NURBS-based isogeometric analysis and the augmented Lagrangian method. *Computational Mechanics*, 49(1):1–20, 2012.

[32] L De Lorenzis, P Wriggers, and TR Hughes. Isogeometric contact: a review. *GAMM-Mitteilungen*, 37(1):85–123, 2014.

[33] EA de Souza Neto, D Peric, and DRJ Owen. *Computational methods for plasticity: theory and applications*. John Wiley & Sons, 2011.

[34] J Deng, F Chen, X Li, C Hu, W Tong, Z Yang, and Y Feng. Polynomial splines over hierarchical T-meshes. *Graphical models*, 70(4):76–86, 2008.

[35] R Dimitri, L De Lorenzis, MA Scott, P Wriggers, RL Taylor, and G Zavarise. Isogeometric large deformation frictionless contact using T-splines. *Computer methods in applied mechanics and engineering*, 269:394–414, 2014.

[36] T Dokken. Locally refined splines. 2010.

[37] T Dokken, T Lyche, and KF Pettersen. Polynomial splines over locally refined box-partitions. *Computer Aided Geometric Design*, 30(3):331–356, 2013.

[38] A Duval, T Elguedj, H Al-Akhras, and F Maurin. abqNURBS: implémentation d'éléments isogéométriques dans Abaqus et outils de pré-et post-traitement dédiés. In *12e Colloque national en calcul des structures*, 2015.

[39] T Elguedj. *Méthodes numériques innovantes pour les simulations robustes en mécanique des solides*. PhD thesis, INSA Lyon; UCBL, 2014.

[40] T Elguedj and TJR Hughes. Isogeometric analysis of nearly incompressible large strain plasticity. *Computer Methods in Applied Mechanics and Engineering*, 268:388–416, 2014.

[41] EJ Evans, MA Scott, X Li, and DC Thomas. Hierarchical T-splines: Analysis-suitability, Bézier extraction, and application as an adaptive basis for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:1–20, 2015.

[42] C Farhat, KG Wang, A Main, S Kyriakides, L-H Lee, K Ravi-Chandar, and T Belytschko. Dynamic implosion of underwater cylindrical shells: experiments and computations. *International Journal of Solids and Structures*, 50 (19):2943–2961, 2013.

[43] DP Flanagan and T Belytschko. Eigenvalues and stable time steps for the uniform strain hexahedron and quadrilateral. *Journal of applied mechanics*, 51(1):35–40, 1984.

[44] C Giannelli and B Jüttler. Bases and dimensions of bivariate hierarchical tensor-product splines. *Journal of Computational and Applied Mathematics*, 239:162–178, 2013.

[45] C Giannelli, B JüTtler, and H Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[46] C Giannelli, B Jüttler, SK Kleiss, A Mantzaflaris, B Simeon, and J Špeh. THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 299:337–365, 2016.

[47] S Govindjee, J Strain, TJ Mitchell, and RL Taylor. Convergence of an efficient local least-squares fitting method for bases with compact support. *Computer Methods in Applied Mechanics and Engineering*, 213:84–92, 2012.

[48] JO Hallquist et al. LS-DYNA theory manual. *Livermore software Technology corporation*, 3:25–31, 2006.

[49] S Hartmann and DJ Benson. Mass scaling and stable time step estimates for isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 102(3-4):671–687, 2015.

[50] RR Hiemstra, F Calabro, D Schillinger, and TJR Hughes. Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:966–1004, 2017.

[51] R Hill. *The mathematical theory of plasticity*, volume 11. Oxford university press, 1998.

[52] T Hirschler, R Bouclier, A Duval, D Crozes, T Elguedj, and J Morlier. Analyse isogéométrique pour les probleèmes d'optimisation de forme des structures coques. 2017.

[53] CAR Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[54] TJR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Dover Publications, 2012.

[55] TJR Hughes, JA Cottrell, and Y Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005. ISSN 00457825.

[56] TJR Hughes, A Reali, and G Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer methods in applied mechanics and engineering*, 199(5-8):301–313, 2010.

[57] KA Johannessen, T Kvamsdal, and T Dokken. Isogeometric analysis using LR B-splines. *Computer Methods in Applied Mechanics and Engineering*, 269: 471–514, 2014.

[58] KA Johannessen, F Remonato, and T Kvamsdal. On the similarities and differences between Classical Hierarchical, Truncated Hierarchical and LR B-splines. *Computer Methods in Applied Mechanics and Engineering*, 291:64–101, 2015.

[59] A Jusuf, T Dirgantara, L Gunawan, and IS Putra. Numerical and Experimental Study of Single-Walled and Double-Walled Columns under Dynamic Axial Crushing. *Journal of Mechanical Engineering*, 9(2):53–72, 2012.

[60] B Jüttler, U Langer, A Mantzaflaris, SE Moore, and W Zulehner. Geometry+ simulation modules: Implementing isogeometric analysis. *PAMM*, 14(1):961–962, 2014.

[61] C Kadapa. *Mixed Galerkin and least-squares formulations for isogeometric analysis*. PhD thesis, PhD thesis, 2014.

[62] P Kagan, A Fischer, and PZ Bar-Yoseph. New B-spline finite element approach for geometrical design and mechanical analysis. *International Journal for Numerical Methods in Engineering*, 41(3):435–458, 1998.

[63] A Kamoulakos. A simple benchmark for impact. *Bench Mark*, pages 31–35, 1990.

[64] H Kang, F Chen, and J Deng. Modified T-splines. *Computer Aided Geometric Design*, 30(9):827–843, 2013.

[65] JY Kim and S Youn. Isogeometric contact analysis using mortar method. *International Journal for Numerical Methods in Engineering*, 89(12):1559–1581, 2012.

[66] G Kiss, C Giannelli, and B Jüttler. Algorithms and data structures for truncated hierarchical B-splines. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 304–323. Springer, 2012.

[67] Y Lai, L Liu, YJ Zhang, J Chen, E Fang, and J Lua. Rhino 3D to Abaqus: A T-Spline Based Isogeometric Analysis Software Framework. In *Advances in Computational Fluid-Structure Interaction and Flow Simulation*, pages 271–281. Springer, 2016.

[68] Y Lai, YJ Zhang, L Liu, X Wei, E Fang, and J Lua. Integrating CAD with Abaqus: a practical isogeometric analysis software platform for industrial applications. *Computers & Mathematics with Applications*, 74(7):1648–1660, 2017.

[69] L Liu, YJ Zhang, TJR Hughes, MA Scott, and TW Sederberg. Volumetric T-spline construction using Boolean operations. *Engineering with Computers*, 30(4):425–439, 2014.

[70] L Liu, YJ Zhang, and X Wei. Handling extraordinary nodes with weighted T-spline basis functions. *Procedia Engineering*, 124:161–173, 2015.

[71] L Liu, YJ Zhang, and X Wei. Weighted T-splines with application in reparameterizing trimmed NURBS surfaces. *Computer Methods in Applied Mechanics and Engineering*, 295:108–126, 2015.

[72] MM Łoś, M Woźniak, M Paszyński, A Lenharth, MA Hassaan, and K Pingali. IGA-ADS: isogeometric analysis FEM using ADS solver. *Computer Physics Communications*, 217:99–116, 2017.

[73] J Lu. Isogeometric contact analysis: Geometric basis and formulation for frictionless contact. *Computer Methods in Applied Mechanics and Engineering*, 200(5-8):726–741, 2011.

[74] ME Matzen and M Bischoff. A weighted point-based formulation for isogeometric contact. *Computer Methods in Applied Mechanics and Engineering*, 308:73–95, 2016.

[75] ME Matzen, T Cichosz, and M Bischoff. A point to segment contact formulation for isogeometric, NURBS based finite elements. *Computer Methods in Applied Mechanics and Engineering*, 255:27–39, 2013.

[76] AP Nagy and DJ Benson. On the numerical integration of trimmed isogeometric elements. *Computer Methods in Applied Mechanics and Engineering*, 284:165–185, 2015.

[77] TN Nguyen. Isogeometric finite element analysis based on Bézier extraction of NURBS and T-splines. Master's thesis, Norges teknisk-naturvitenskapelige universitet, Fakultet for ingeniørvitenskap og teknologi, Institutt for konstruksjonsteknikk, 2011.

[78] VP Nguyen and S Bordas. Extended isogeometric analysis for strong and weak discontinuities. In *Isogeometric methods for numerical simulation*, pages 21–120. Springer, 2015.

[79] VP Nguyen, C Anitescu, S Bordas, and T Rabczuk. Isogeometric analysis: an overview and computer implementation aspects. *Mathematics and Computers in Simulation*, 117:89–116, 2015.

[80] N Nguyen-Thanh, J Kiendl, H Nguyen-Xuan, R Wüchner, KU Bletzinger, Y Bazilevs, and T Rabczuk. Rotation free isogeometric thin shell analysis using PHT-splines. *Computer Methods in Applied Mechanics and Engineering*, 200(47):3410–3424, 2011.

[81] L Olovsson, M Unosson, and K Simonsson. Selective mass scaling for thin walled structures modeled with tri-linear solid elements. *Computational Mechanics*, 34(2):134–136, 2004.

[82] L Olovsson, K Simonsson, and M Unosson. Selective mass scaling for explicit finite element analyses. *International Journal for Numerical Methods in Engineering*, 63(10):1436–1445, 2005.

[83] M Pagani, S Reese, and U Perego. Computationally efficient explicit nonlinear analyses using reduced integration-based solid-shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 268:141–159, 2014.

[84] KC Park. Practical aspects of numerical time integration. *Computers & Structures*, 7(3):343–353, 1977.

[85] MS Pauletti, M Martinelli, N Cavallini, and P Antolin. Igatools: An isogeometric analysis library. *SIAM Journal on Scientific Computing*, 37(4): C465–C496, 2015.

[86] L Piegl and W Tiller. *The NURBS Book (2Nd Ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997. ISBN 3-540-61545-8.

[87] KG Rakvåg, T Børvik, I Westermann, and OS Hopperstad. An experimental study on the deformation and fracture modes of steel projectiles during impact. *Materials & Design*, 51:242–256, 2013.

[88] KG Rakvåg, T Børvik, and OS Hopperstad. A numerical study on the deformation and fracture modes of steel projectiles during Taylor bar impact tests. *International Journal of Solids and Structures*, 51(3-4):808–821, 2014.

[89] PJ Roache. Computational Fluid Dynamics (Hermosa, Albuquerque, 1976; Mir, Moscow, 1980). *Google Scholar*, 2006.

[90] D Rypl and B Patzák. From the finite element analysis to the isogeometric analysis in an object oriented computing environment. *Advances in Engineering Software*, 44(1):116–125, 2012.

[91] D Schillinger, L Dede, MA Scott, JA Evans, MJ Borden, E Rank, and TJR Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 249:116–150, 2012.

[92] D Schillinger, JA Evans, A Reali, MA Scott, and TJR Hughes. Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Computer Methods in Applied Mechanics and Engineering*, 267:170–232, 2013.

[93] D Schillinger, SJ Hossain, and TJR Hughes. Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 277:1–45, 2014.

[94] MA Scott, X Li, TW Sederberg, and TJR Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213:206–222, 2012.

[95] TW Sederberg, J Zheng, A Bakenov, and A Nasri. T-splines and T-NURCCs. In *ACM transactions on graphics (TOG)*, volume 22, pages 477–484. ACM, 2003.

[96] TW Sederberg, DL Cardon, GT Finnigan, NS North, J Zheng, and T Lyche. T-spline simplification and local refinement. In *ACM transactions on graphics (TOG)*, volume 23, pages 276–283. ACM, 2004.

[97] I Temizer and C Hesch. Hierarchical NURBS in frictionless contact. *Computer Methods in Applied Mechanics and Engineering*, 299:161–186, 2016.

[98] I Temizer, P Wriggers, and TJR Hughes. Contact treatment in isogeometric analysis with NURBS. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):1100–1112, 2011.

[99] I Temizer, P Wriggers, and TJR Hughes. Three-dimensional mortar-based frictional contact treatment in isogeometric analysis with NURBS. *Computer Methods in Applied Mechanics and Engineering*, 209:115–128, 2012.

[100] SP Timoshenko and J Goodyear. Elasticity theory. *M: Science*, 1975.

[101] AV Vuong, C Giannelli, B Jüttler, and B Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49):3554–3567, 2011.

[102] P Wang, J Xu, J Deng, and F Chen. Adaptive isogeometric analysis using rational PHT-splines. *Computer-Aided Design*, 43(11):1438–1448, 2011.

[103] X Wang and X Qian. An optimization approach for constructing trivariate B-spline solids. *Computer-Aided Design*, 46:179–191, 2014.

[104] X Wei, YJ Zhang, L Liu, and TJR Hughes. Truncated T-splines: Fundamentals and methods. *Computer Methods in Applied Mechanics and Engineering*, 316:349–372, 2017.

[105] JH Wilkinson. The algebraic eigenvalue problem. In *Handbook for Automatic Computation, Volume II, Linear Algebra*. Springer-Verlag New York, 1971.

[106] G Willkenmuller and K Kayvantash. Radioss theoretical manual. *MECALOG archives*, 1999.

[107] G Xu, M Li, B Mourrain, T Rabczuk, J Xu, and S Bordas. Constructing IGA-suitable planar parameterization from complex CAD boundary by domain partition and global/local optimization. *Computer Methods in Applied Mechanics and Engineering*, 328:175–200, 2018.

[108] T Xuan Duong, L De Lorenzis, and RA Sauer. A segmentation-free isogeometric extended mortar contact method. *CoRR*, abs/1712.01179, 2017.

[109] L Zhang and D Eyheramendy. Une approche isogéométrique pour la thermomécanique en grandes transformations dans un contexte multiphysique à objets. In *12e colloque national en calcul des structures*, 2015.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : OCCELLI DATE de SOUTENANCE : 29/11/2018

Prénoms : Matthieu

TITRE : Explicit dynamics IGA. LR B-Splines implementation in the Radioss solver

NATURE : Doctorat Numéro d'ordre : 2018LYSEI102

Ecole doctorale : Mécanique, Energétique, Génie civil, Acoustique (MEGA) n°162

Spécialité : Mécanique – Génie mécanique – Génie civil

RESUME :

L'analyse isométrique s'est révélée être un outil très prometteur pour la conception et l'analyse. Une tâche difficile consiste toujours à faire passer l'IGA de concept à un outil de conception pratique pour l'industrie et ce travail contribue à cet effort. Ce travail porte sur l'implémentation de l'IGA dans le solveur explicite Altair Radioss afin de répondre aux applications de simulation de crash et d'emboutissage. Pour cela, les ingrédients nécessaires à une intégration native de l'IGA dans un code éléments finis traditionnel ont été identifiés et adaptés à l'architecture de code existante. Un élément solide B-Spline et NURBS a été développé dans Altair Radioss. Les estimations heuristiques des pas de temps élémentaires ou nodaux sont explorées pour améliorer l'efficacité des simulations et garantir leur stabilité. Une interface de contact existante a été étendue afin de fonctionner de manière transparente avec les éléments finis NURBS et de Lagrange. Un raffinement local est souvent nécessaire pour la bonne représentation de champs non linéaires tels que les champs de déformations plastiques. Une analyse est faite en termes de compatibilité pour l'analyse et de mise en œuvre pour plusieurs bases de fonctions Spline telles que les Hierarchical B-Splines, les Truncated Hierarchical B-Splines, les T-Splines et les Locally Refined B-Splines (LR B-Splines). Les LR B-Splines sont implémentés. Un schéma de raffinement est proposé et définit un sous-ensemble de raffinements adapté à leur utilisation au sein de Radioss. Le processus de raffinement d'un maillage initialement grossier et régulier est développé au sein du solveur. Il permet à l'utilisateur d'établir du raffinement local par un ensemble d'instructions à fournir dans le jeu de donnée de la simulation. La solution globale est validée sur des cas tests industriels, pour des cas de validation classiquement utilisés pour les codes industriels comme l'emboutissage et les tests de chute.

MOTS-CLÉS : Solveur industriel, Analyse IsoGéométrique, Approche volumique, Raffinement local, Dynamique explicite

Laboratoire (s) de recherche : Laboratoire de Mécanique des Contacts et des Solides
CNRS UMR5259 – Université de Lyon, INSA de Lyon
20, avenue Albert Einstein
69621 Villeurbanne Dedex FRANCE

Directeur de thèse: ELGUEDJ Thomas

Président de jury : BARANGER Thouraya

Composition du jury :

BARANGER Thouraya	Professeur, UCBL	Examinateur
BOUABDALLAH Salim	Docteur, Altair Engineering	Examinateur
BOUCLIER Robin	Maître de Conférences, INSA Toulouse	Invité
BREITKOPF Piotr	HDR, UTC Compiègne	Examinateur
ELGUEDJ Thomas	Professeur, INSA de Lyon	Directeur de thèse
EYHERAMENDY Dominique	Professeur, Ecole Centrale Marseille	Rapporteur
REALI Alessandro	Professeur, University of Pavia	Rapporteur