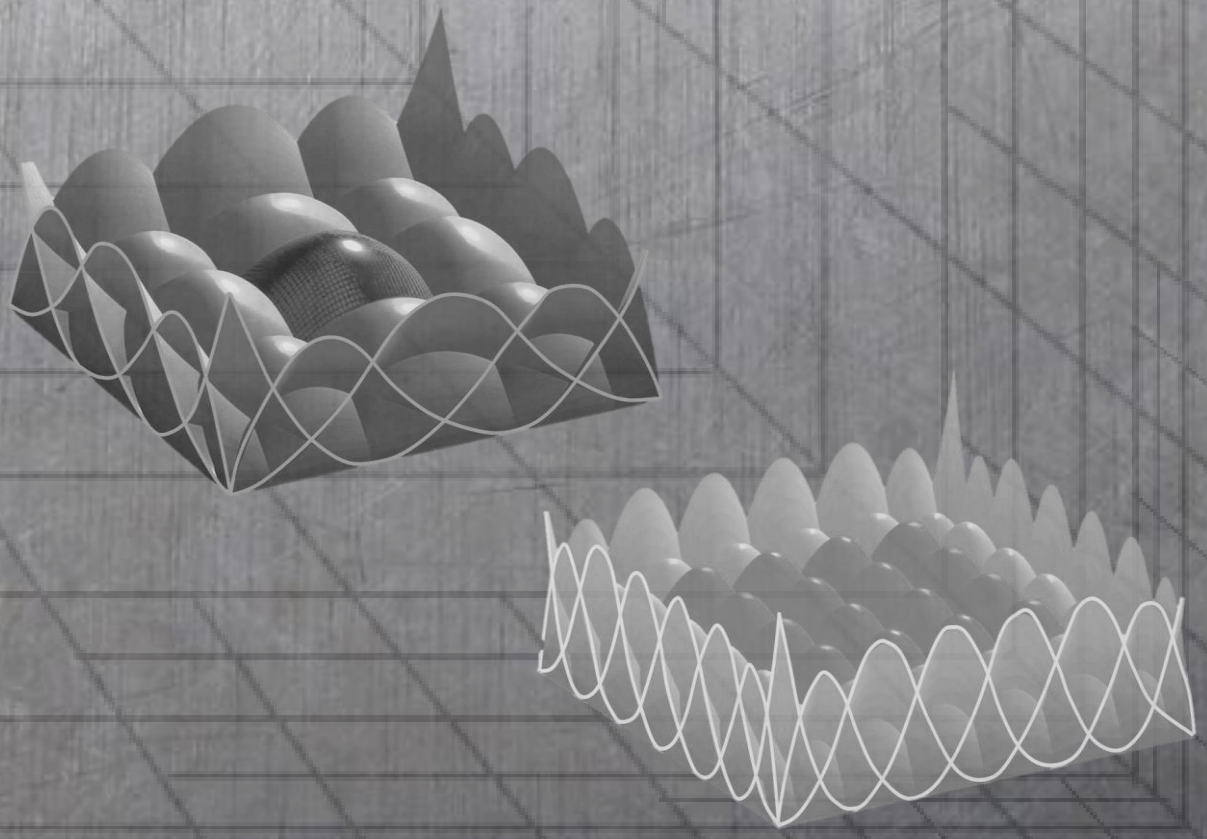
NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF CIVIL ENGINEERING

ADAPTIVE HIERARCHICAL REFINEMENT IN ISOGEOMETRIC ANALYSIS

DIPLOMA THESIS



ANTONIOS I. IAKOVOS

Athens
November 2015

Supervisor:
Prof. Manolis Papadrakakis

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF CIVIL ENGINEERING

Adaptive Hierarchical Refinement in Isogeometric Analysis

Antonios I. Iakovos

Diploma Thesis

Supervisor: Prof. Manolis Papadrakakis

Athens, November 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Προσαρμοστική Ιεραρχική Βελτίωση της Προσομοίωσης με τη Μέθοδο της Ισογεωμετρικής Ανάλυσης

Αντώνιος Ι. Ιάκωβος

Διπλωματική Εργασία

Επιβλέπων: Μανόλης Παπαδρακάκης, Καθηγητής

Αθήνα, Νοέμβριος 2015

Στην οικογένεια μου

ACKNOWLEDGEMENTS

The present thesis has been completed after two years of hard but very interesting work. There was no easy way, sacrifices have been made but the outcome fills me with hope. During this interesting journey of knowledge there were many ups and downs. Nothing would be possible without the people that encouraged and supported me in every step of this work. They were always there for me to endure my effort so I would like to apologize to them for what they went through with me all this time and express them my sincere gratitude. I hope the result will meet their expectations.

I would also like to express my sincere gratitude to my supervisor Professor Manolis Papadrakakis, who gave me the opportunity to make this journey of knowledge. It was him who introduced me in computational mechanics first as teacher and then as supervisor of this thesis. His guidance, patience, advice and encouragement were always crucial to carry on and complete successfully this thesis that fills me with hope. The countless discussions at his office helped me mature and think as a young researcher. In this sense I would also like to thank him for giving me the opportunity to attend COMPDYN 2015 and UNCECOMP 2015 in Crete. It was an amazing experience that expanded my knowledge on computational mechanics and civil engineering in general.

Another person I would like to thank is PhD candidate Panagiotis Karakitsios who welcomed me into the world of isogeometric analysis. I am grateful for the opportunity he gave me and the help he offered in this two years journey in computational mechanics.

I would also like to thank all my friends, old and new, for always giving me hope. Moreover I would like to thank my friend and fellow-researcher Dimitris Karras for his help and the countless discussions that led to this result. I am also grateful for all my fellows from university who along with Dimitris and me shared the same interest and time to cope with other isogeometric analysis subjects.

Last but not least, I would like to thank my family for all these years of support and understanding. Without them nothing would be possible. Their unconditional love, patience and encouragement helped me not only to evolve as a scientist but as a human as well.

Antonios I. Iakovos

Athens, November 2015

ABSTRACT

The aim of the present thesis is to study in depth adaptive hierarchical refinement methods in isogeometric analysis. Hierarchical refinement is at this point a very interesting and promising research topic in the field of isogeometric analysis and computational mechanics in general. It is a research subject which emerged in the last few years to enrich isogeometric analysis with procedures already known and used in finite element theory. It is obvious that isogeometric adaptive refinement couldn't have been applied without any progress in local refinement. This progress was firstly presented by introducing T-Splines. But the widespread technologies of B-Splines and NURBS made the need of their enrichment with local refinement properties imperative. In this sense A.Vuong proposed in 2011 a method based on CAD local hierarchical geometry refinement suitable for analysis. Since then many other methods are proposed to take full advantage of Vuong's method in isogeometric analysis. Hierarchical refinement is a technique of local enrichment of the approximation space by refining shape functions. For this reasons the term ''hierarchical'' refers only to the relation between shape functions and not to the formulation of stiffness matrices. In the present thesis some of the isogeometric adaptive methods are presented in order to compare them in a theoretical basis. Their properties, advantages, disadvantages and applying techniques are fully explained. The final goal is to feature their special characteristics which can be combined, leading to enhancement of problem solution. To quantify some general properties of the described refinement methods, an algorithm using adaptive hierarchical refinement is also presented. Algorithm results are compared with results derived from uniform refinement to highlight adaptive refinement's advantages.

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας διπλωματικής είναι η εμβάθυνση στις μεθόδους προσαρμοστικής ιεραρχικής πύκνωσης των συναρτήσεων σχήματος στα πλαίσια της ισογεωμετρικής ανάλυσης. Η ιεραρχική επαναδιακριτοποίηση, είναι αυτήν την στιγμή ένα απο τα πλέον ανερχόμενα θέματα έρευνας στο πεδίο της ισογεωμετρικής ανάλυσης και της υπολογιστικής μηχανικής γενικότερα. Είναι ένα σχετικά πρόσφατο ερευνητικό πεδίο που στοχέυει στον εμπλουτισμό της ισογεωμετρικής ανάλυσης με ήδη γνωστούς μηχανισμούς από την μέθοδο των πεπερασμένων στοιχείων. Είναι σαφές πως θα ήταν αδύνατο να πραγματοποιηθεί προσαρμοστική πύκνωση χωρίς την δυνατότητα τοπικής επαναδιακριτοποίησης ισογεωμετρικών δικτύων. Το κενό της τοπικής πύκνωσης στην ισογεωμετρική ήρθαν αρχικά να καλύψουν οι T-Splines. Η έρευνα όμως δεν σταμάτησε εκεί, καθώς υπήρχε η ανάγκη διατύπωσης μεθόδων βασισμένων στις ιδιαίτερα διαδεδομένες B-Splines και NURBS. Σ' αυτό το πλαίσιο το 2011 η A.Vuong εκμεταλλευόμενη την ιεραρχική πύκνωση γεωμετρικών δικτύων (CAD) πρότεινε μία μέθοδο για την αξιοποίηση τους στην ανάλυση. Έκτοτε έχουν παρουσιαστεί νέες μέθοδοι για την αξιοποίηση της πρότασης της Vuong στην ισογεωμετρική ανάλυση. Η ιεραρχική πύκνωση είναι μια τεχνική τοπικού εμπλουτισμού των συναρτήσεων σχήματος για την βελτίωση της λύσης. Να σημειωθεί εδώ πως ο όρος ιεραρχική αναφέρεται αποκλειστικά στην ιεραρχική διατύπωση των συναρτήσεων σχήματος και όχι στην ιεραρχική μόρφωση του μητρώου στιβαρότητας. Στην παρούσα διπλωματική γίνεται παρουσίαση των μεθόδων προσαρμοστικής επαναδιακριτοποίησης σε μια προσπάθεια να υπάρξει μια αρχική θεωρητική και ποιοτική σύγκριση για την αξιολόγηση τους. Παρουσιάζονται οι ιδιότητες τους , τα πλεονεκτήματα , τα μειονεκτήματα και ο τρόπος εφαρμογής τους. Τελικός στόχος η ανάδειξη των ιδιαίτερων στοιχείων τους που συνδυαστικά μπορούν να βελτιώσουν την επίλυση προβλημάτων. Για τον σκοπό αυτό παρουσιάζεται και αλγόριθμος επίλυσης προβλημάτων με την εφαρμογή της προσαρμοστικής ιεραρχικής πύκνωσης. Σκοπός η σύγκριση με την κλασική μέθοδο επαναδιακριτοποίησης της ισογεωμετρικής ανάλυσης (εισαγωγή κόμβων) .

ΕΚΤΕΤΑΜΕΝΗ ΠΕΡΙΛΗΨΗ

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΙΣΟΓΕΩΜΕΤΡΙΚΗ ΑΝΑΛΥΣΗ

Η ισογεωμετρική ανάλυση είναι μια σχετικά νέα μέθοδος. Προτάθηκε από τους Cottrell, Hughes, Bazilevs το 2005 και αποτελεί από τότε ένα πεδίο εξαιρετικού ερευνητικού ενδιαφέροντος. Η λογική της ισογεωμετρικής θεμελιώνεται στην ιδέα της γεφύρωσης δύο διαφορετικών τεχνολογιών, της CAD και CAE. Για τον σκοπό αυτό οι συναρτήσεις σχήματος της ψηφιακής σχεδίασης χρησιμοποιούνται ως συναρτήσεις σχήματος και στην ανάλυση. Αποτέλεσμα αυτής της ιδέας είναι πως στην ισογεωμετρική η ανάλυση γίνεται πάντα με την ακριβή αρχική γεωμετρία, ακόμα και σε πολύ αραιές διακριτοποιήσεις. Οι πιο εδραιωμένες συναρτήσεις σχήματος για την ισογεωμετρική ανάλυση είναι οι B-Splines κι οι NURBS, οι οποίες λόγω των καλών τους ιδιοτήτων μπορούν να χρησιμοποιηθούν χωρίς σημαντικές τροποποιήσεις στην επίλυση προβλημάτων σε κώδικα για πεπερασμένα στοιχεία.

ΧΩΡΟΙ PHYSICAL, PARAMETER ΚΑΙ INDEX

Τα στοιχεία που χρησιμοποιούνται στην ισογεωμετρική ανάλυση με NURBS είναι ισοπαραμετρικά και μπορούν να παρασταθούν σε τρεις χώρους.

- Πραγματικός χώρος (physical space)
- Παραμετρικός χώρος (parameter space)
- Χώρος των δεικτών (index space)

Ο physical space είναι ο χώρος που το αντικείμενο έχει την πραγματική του μορφή στον οποίο σχεδιάζεται με τα προγράμματα CAD ενώ ο parameter space είναι ο χώρος στον οποίο γίνεται η διακριτοποίηση του φορέα και η αριθμητική ολοκλήρωση για τη μόρφωση του μητρώου στιβαρότητας. Ο χώρος των δεικτών (index space) επιτελεί βοηθητικό ρόλο στις συναρτήσεις NURBS. Χρησιμοποιείται για τον προσδιορισμό των παραμετρικών συντεταγμένων των control points και για την καλύτερη εποπτεία του Knot Value Vector.

B-SPLINES

Οι συναρτήσεις NURBS είναι μια γενικότερη, ρητή μορφή με βάρη, των B-Splines. Ουσιαστικά αποτελούν προβολές τους επί επιπέδου. Απαιτούνται n+p+1 μη μειούμενες παραμετρικές συντεταγμένες που ορίζουν το knot value vector, $\Xi = \left\{ \xi_1, \xi_2, \cdots \xi_{n+p}, \xi_{n+p+1} \right\}$ όπου n είναι ο αριθμός των συναρτήσεων βάσεως και p ο πολυωνυμικός βαθμός τους. Στο knot value vector αντιστοιχίζεται ένα knot vector, στο οποίο αμελούνται οι επαναλήψεις των ίδιων τιμών και αποθηκεύονται μόνο διακριτά μεταξύ τους στοιχεία.Με βάση το knot

value vector ορίζονται οι συναρτήσεις βάσης με τον αναδρομικό αλγόριθμο Cox de Boor, ως εξής:

$N_{i,0}(\xi) = \begin{cases} 1, & \text{if } \xi_i \le \xi < \xi_{i+1} \\ 0, & \text{otherwise} \end{cases}$, p=0 και $N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi)$, p>0

Η k παράγωγός τους προσδιορίζεται από τη σχέση: $\frac{d^k}{d^k \xi} N_{i,p}(\xi) = \frac{p!}{(p-k)!} \sum_{j=0}^{k} a_{k,j} N_{i+j,p-k}(\xi)$

Οι B-Splines έχουν την ιδιότητα του πλήρους τανυστικού γινομένου (full tensor product). Με αυτό τον τρόπο μπορούν να δημιουργηθούν οι συναρτήσεις σχήματος, καθώς αποτελούν συνδυασμό των επιμέρους παραμετρικών συναρτήσεων βάσης. Η i B-Spline συνάρτηση σχήματος ορίζεται ως:

$R_i^p(\xi) = N_{i,p}(\xi)$ σε μονοδιάστατα προβλήματα,

$R_{i,j}^{p,q}(\xi,\eta) = N_{i,p}(\xi) M_{j,q}(\eta)$ σε διδιάστατα και

$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta)$ σε τριδιάστατα.

Τις B-Splines χαρακτηρίζουν οι παρακάτω ιδιότητες :

- Είναι μη μηδενικές σε περιορισμένο διάστημα (support): $N_{i,p}(\xi) = 0 \quad \forall \xi \notin \left[\xi_i, \xi_{i+p+1}\right)$
- Σε κάθε διάστημα $\left[\xi_j, \xi_{j+1}\right)$ το πολύ (p+1) εκ των συναρτήσεων $N_{i,p}$ είναι μη μηδενικές, και οι πιθανές μη μηδενικές είναι οι: $N_{j-p,p}, \ldots, N_{j,p}$.
- Κάθε συνάρτηση μοιράζεται μέρη του μη μηδενικού διαστήματός της με 2p άλλες.
- Οι συναρτήσεις είναι παντού μη αρνητικές $N_{i,p}(\xi) \ge 0$
- Το άθροισμα των συναρτήσεων σχήματος σε οποιοδήποτε σημείο είναι 1. $\sum_{i=1}^{n} N_{i,p}(\xi) = 1$
- Οι συναρτήσεις έχουν συνέχεια C^{p-k} πάνω σε έναν κόμβο πολλαπλότητας k και είναι απείρως διαφορίσιμες στο εσωτερικό του διαστήματος μεταξύ των κόμβων.
- Εκτός απ' την περίπτωση των συναρτήσεων σχήματος για p=0, όλες οι άλλες έχουν ακριβώς ένα μέγιστο.

Τα control points στον παραμετρικό χώρο, βρίσκονται στο μέσον του support της συναρτήσεως βάσεως που αντιστοιχούν. Με βάση την ιδιότητα (1) των B-Spline, οι παραμετρικές συντεταγμένες του i control point στον άξονα ξ ορίζονται ως:

$$\xi_{CP_i} = 0.5\left(\xi_{i+\frac{p}{2}} + \xi_{i+\frac{p}{2}+1}\right).$$

Οι γεωμετρίες των B-Splines μπορεί να είναι καμπύλες (1D), επιφάνειες (2D) ή όγκοι (3D) και ορίζονται ως το άθροισμα των συναρτήσεων σχήματος πολλαπλασιασμένων με τις συντεταγμένες των αντίστοιχων control point. Μια γεωμετρία B-Spline ορίζεται ως

$$(x,y,z) = C(\xi) = \underset{(1xn)}{\left\{N_{i,p}(\xi)\right\}^T} \underset{(nx3)}{\left\{P_i\right\}} = \sum_{i=1}^{n} N_{i,p}(\xi) \underset{(1x3)}{\left\{P_i\right\}} \quad , \quad \xi_1 \le \xi \le \xi_{n+p+1} \text{ για καμπύλη,}$$

$$(x,y,z) = S(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}(\xi,\eta) \underset{(1x3)}{\left\{P_{i,j,k}\right\}} = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) N_{j,q}(\eta) \underset{(1x3)}{\left\{P_{i,j}\right\}} \text{ για επιφάνεια}$$

και ένας όγκος ως

$$(x,y,z) = V(\xi,\eta,\zeta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) \underset{(1x3)}{\left\{P_{i,j,k}\right\}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi) N_{j,q}(\eta) N_{k,r}(\xi) \underset{(1x3)}{\left\{P_{i,j,k}\right\}}$$

Οι καμπύλες B-Splines, και κατά επέκταση οι επιφάνειες και οι όγκοι, έχουν μια σειρά από θεμιτές ιδιότητες, απόρροια των ιδιοτήτων των συναρτήσεων σχήματος από τις οποίες φτιάχνονται.

- Είναι μια γενίκευση των καμπυλών Bezier.
- Η C(ξ) είναι μια τμηματική πολυωνυμική καμπύλη.
- Το πρώτο και το τελευταίο control point βρίσκονται πάνω στη καμπύλη και μάλιστα είναι το πρώτο και το τελευταίο σημείο της, $C(\xi_1) = P_1$ και $C(\xi_{n+p+1}) = P_n$.
- Οι καμπύλες B-Spline έχουν την ιδιότητα το τμήμα τους $\left[\xi_i, \xi_{i+1}\right)$ να περιέχεται εντός του “Convex Hull” των control point $P_{i-p}, \ldots, P_i$.
- Η μετακίνηση ενός control point P αλλάζει επηρεάζει μόνο την περιοχή κοντά στο control point.
- Το πολύγωνο των control points είναι μια γραμμική προσέγγιση της καμπύλης
- Ένας οποιοδήποτε συνδυασμός γραμμικών μετασχηματισμών μπορεί να εφαρμοστεί στην καμπύλη με την εφαρμογή του συνδυασμού αυτού στα control points.
- Κανένα επίπεδο δεν έχει περισσότερες τομές με την καμπύλη από ότι με το πολύγωνο των control points (Σε διδιάστατες καμπύλες, καμία γραμμή δεν έχει περισσότερες τομές με την καμπύλη από ότι με το πολύγωνο των control points).
- C(ξ) είναι γραμμικός συνδυασμός των $N_{i,p}(\xi)$, επομένως η συνέχεια και η παραγωγισιμότητα της καμπύλης προκύπτουν από τις αντίστοιχες συναρτήσεις βάσης.
- Είναι δυνατό και πολλές φορές χρήσιμο να χρησιμοποιήσουμε πολλαπλά control points με τις ίδιες καρτεσιανές συντεταγμένες.

Όλες οι παραπάνω ιδιότητες γενικεύονται στις 2D επιφάνειες και σε 3D στερεά σώματα.

NURBS

Η καμπύλη NURBS θεωρείται ως προβολή της αντίστοιχης καμπύλης B-Spline με control points $P_i^w = (X_i, Y_i, w_i)$ σε ένα συγκεκριμένο επίπεδο με control points $P_i = \left(\frac{X_i}{w_i}, \frac{Y_i}{w_i} \right)$ και βάρος w_i. Στη γενικότερη περίπτωση, μια γεωμετρία NURBS μπορεί να προβληθεί από το ρητό χώρο d-διαστάσεων στο χώρο των B-Splines των (d+1) διαστάσεων και αντίστροφα.

Η προβολή των control points των NURBS στο χώρο των B-Splines γίνεται με τον πολλαπλασιασμό του control point με το αντίστοιχο βάρος και με διαίρεση για την αντίστροφη προβολή. Τα control points που ανήκουν στις B-Splines ονομάζονται προβολικά (projective) και συμβολίζονται με P_i^w .

$$(3D - \text{ρητός χώρος}) \qquad \underset{(nx3)}{\{P\}} = \underset{(nx3)}{\{X_i, Y_i, Z_i\}} \text{ , με βάρος } w_i \xrightleftharpoons[/w_i]{\cdot w_i}$$

$$(4D - \text{ μη ρητός χώρος}) \quad \underset{(nx4)}{\{P^w\}} = \underset{(nx4)}{\{w_i X_i, w_i Y_i, w_i Z_i, w_i\}}$$

Για να οριστούν οι συναρτήσεις NURBS πρέπει να οριστεί πρώτα μια συνάρτηση βάρους. Οι συναρτήσεις σχήματος NURBS ορίζονται με την προβολή των συναρτήσεων B-Splines ανάλογα με τη διάσταση του χώρου που δουλεύουμε ως:

1D:

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

2D:

$$W(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta)\, w_{i,j} \,, \qquad R_{i,j}^{p,q}(\xi,\eta) = \frac{w_{i.j}\, N_{i,p}(\xi) M_{j,q}(\eta)}{W(\xi,\eta)}$$

3D:

$$W(\xi,\eta,\zeta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta)\, w_{i,j,k}$$

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = \frac{w_{i.j,k}\, N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta)}{W(\xi,\eta,\zeta)}$$

Οι παράγωγοι των NURBS προκύπτουν από την παραγώγιση της συνάρτησης τους

1D:

$$\frac{d}{d\xi}R_i^p(\xi)=\frac{\left(\frac{d}{d\xi}N_{i,p}(\xi)\right)W(\xi)-N_{i,p}(\xi)\left(\frac{d}{d\xi}W(\xi)\right)}{\left(W(\xi)\right)^2}w_i$$

2D:

$$\frac{\partial}{\partial\xi}R_{i,j}^{p,q}(\xi,\eta)=\frac{\left(\frac{\partial}{\partial\xi}N_{i,p}(\xi)\right)W(\xi,\eta)-N_{i,p}(\xi)\left(\frac{\partial}{\partial\xi}W(\xi,\eta)\right)}{\left(W(\xi,\eta)\right)^2}w_{i,j}M_{j,q}(\eta)$$

$$\frac{\partial}{\partial\eta}R_{i,j}^{p,q}(\xi,\eta)=\frac{\left(\frac{\partial}{\partial\eta}M_{j,q}(\eta)\right)W(\xi,\eta)-M_{j,q}(\eta)\left(\frac{\partial}{\partial\eta}W(\xi,\eta)\right)}{\left(W(\xi,\eta)\right)^2}w_{i,j}N_{i,p}(\xi)$$

και ομοίως για την 3D περίπτωση.

Αντίστοιχα με τον ορισμό των γεωμετριών B-Splines, οι γεωμετρίες μπορεί να είναι καμπύλες (1D), επιφάνειες (2D) και όγκοι (3D) και ορίζονται ως άθροισμα των συναρτήσεων σχήματος πολλαπλασιασμένων με τις συντεταγμένες των αντίστοιχων control points. Μια γεωμετρία NURBS ορίζεται ως:

Καμπύλη

$$(x,y,z)=C(\xi)=\underset{(1xn)}{\left\{N_{i,p}(\xi)\right\}^T}\underset{(n\,x\,3)}{\left\{P_i\right\}}=\sum_{i=1}^{n}N_{i,p}(\xi)\underset{(1x3)}{\left\{P_i\right\}}\quad,\quad \xi_1\le\xi\le\xi_{n+p+1}$$

Επιφάνεια

$$(x,y,z)=S(\xi,\eta)=\sum_{i=1}^{n}\sum_{j=1}^{m}R_{i,j}^{p,q}(\xi,\eta)\underset{(1x3)}{\left\{P_{i,j,k}\right\}}=\sum_{i=1}^{n}\sum_{j=1}^{m}N_{i,p}(\xi)N_{j,q}(\eta)\underset{(1x3)}{\left\{P_{i,j}\right\}}$$

Όγκος

$$(x,y,z)=V(\xi,\eta,\zeta)=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)\underset{(1x3)}{\left\{P_{i,j,k}\right\}}=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}N_{i,p}(\xi)N_{j,q}(\eta)N_{k,r}(\xi)\underset{(1x3)}{\left\{P_{i,j,k}\right\}}$$

Λόγω του τρόπου δημιουργίας των NURBS από τις B-Splines, οι NURBS κληρονομούν όλες τις καλές ιδιότητες των B-Splines που προαναφέρθηκαν.

ΜΗΤΡΩΟ ΣΤΙΒΑΡΟΤΗΤΑΣ

Στην ισογεωμετρική ανάλυση ακολουθείται η ίδια λογική με τα πεπερασμένα στοιχεία για τη μόρφωση του μητρώου στιβαρότητας. Ο φορέας διακριτοποιείται σε στοιχεία που δημιουργούνται από τα knot vectors και χρησιμοποιείται αριθμητική ολοκλήρωση κατά Gauss για τον αριθμητικό υπολογισμό το μητρώο στιβαρότητας. Αν ο μεγαλύτερος εκ των βαθμών των πολυωνύμων στους παραμετρικούς άξονες είναι p, τότε αριθμός των Gauss points κατά τον άξονα σε κάθε στοιχείο είναι:

$$n_{GP}^{perKnotSpan} = \begin{cases} p, \text{ για 1-}\Delta \text{ προβλήματα} \\ p+1, \text{ για 2-}\Delta \text{ ,3-}\Delta \text{ προβλήματα} \end{cases}$$

Οι συντεταγμένες ξ^R και τα βάρη w_ξ^R των Gauss points συναρτήσει του αριθμού τους υπάρχουν στη βιβλιογραφία για ένα διάστημα ολοκλήρωσης (-1,1). Ανάγοντάς τις στο παραμετρικό διάστημα του κάθε στοιχείου, προκύπτουν οι παραμετρικές συντεταγμένες και τα αντίστοιχα βάρη:

$$\xi = \frac{(\xi_{i+1} - \xi_i)\xi^R + (\xi_{i+1} + \xi_i)}{2}$$

$$w^{GP\xi} = \frac{(\xi_{i+1} - \xi_i)}{2} w_\xi^R$$

Θεωρώντας γραμμική ελαστικότητα, μπορεί να μορφωθεί το μητρώο ελαστικότητας [E] για περιπτώσεις μονοδιάστατης (δικτυώματος), διδιάστατης (επίπεδης έντασης ή επίπεδης παραμόρφωσης) και τριδιάστατης ελαστικότητας.

Truss

$$\underset{(1x1)}{\left[E_{Truss}\right]} = E$$

Plane Stress

$$\underset{(3x3)}{\left[E_{Plane\ Stress}\right]} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}$$

Plane Strain

$$\underset{(3x3)}{\left[E_{Plane\ Strain}\right]} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}$$

3D Elasticity

$$\underset{(6x6)}{\left[E_{\text{3D Elasticity}}\right]}=\frac{E}{(1-\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & \nu & 0 & 0 & 0\\ \nu & 1-\nu & \nu & 0 & 0 & 0\\ \nu & \nu & 1-\nu & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2}\end{bmatrix}$$

Ο Ιακωβιανός πίνακας και ο αντίστροφός του για τις μεταβάσεις από τον πραγματικό στον παραμετρικό χώρο είναι ο ακόλουθος για τις 3 περιπτώσεις ελαστικότητας:

1D Elasticity	2D Elasticity	3D Elasticity
$\underset{(1x1)}{[J]}=\underset{(1x1)}{\left[\frac{dx}{d\xi}\right]}=\underset{(1xn)}{\{N_\xi\}^T}\underset{(nx1)}{\{P\}}$	$\underset{(2x2)}{[J]}=\underset{(2x2)}{\begin{bmatrix}\frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi}\\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta}\end{bmatrix}}=\underset{(2xn)}{\begin{Bmatrix}\{N_\xi\}^T\\ \{N_\eta\}^T\end{Bmatrix}}\underset{(nx2)}{\{P\}}$	$\underset{(3x3)}{[J]}=\underset{(3x3)}{\begin{bmatrix}\frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} & \frac{\partial z}{\partial\xi}\\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} & \frac{\partial z}{\partial\eta}\\ \frac{\partial x}{\partial\zeta} & \frac{\partial y}{\partial\zeta} & \frac{\partial z}{\partial\zeta}\end{bmatrix}}=\underset{(3xn)}{\begin{Bmatrix}\{N_\xi\}^T\\ \{N_\eta\}^T\\ \{N_\zeta\}^T\end{Bmatrix}}\underset{(nx3)}{\{P\}}$
$\underset{(1x1)}{[J]^{-1}}=\frac{1}{\underset{(1x1)}{[J]}}=\frac{1}{\det([J])}$	$\underset{(2x2)}{[J]^{-1}}=\begin{bmatrix}J_{11}^* & J_{12}^*\\ J_{21}^* & J_{22}^*\end{bmatrix}=\frac{1}{\det[J]}\begin{bmatrix}J_{22} & -J_{12}\\ -J_{21} & J_{11}\end{bmatrix}$ where $\det[J]=J_{11}J_{22}-J_{21}J_{12}$	$\underset{(3x3)}{[J]^{-1}}=\begin{bmatrix}J_{11}^* & J_{12}^* & J_{13}^*\\ J_{21}^* & J_{22}^* & J_{23}^*\\ J_{31}^* & J_{32}^* & J_{33}^*\end{bmatrix}$

Τα μητρώα παραμορφώσεως για μονοδιάστατη ελαστικότητα **1D** είναι:
Τα επιμέρους μητρώα παραμόρφωσης:

$$\underset{(1x1)}{[B_1]}=\left[\frac{1}{J_{11}}\right]\text{ και }\underset{(1xn)}{[B_2(\xi)]}=\underset{(1xn)}{\{R_{,\xi}\}^T}.$$

Το ολικό μητρώο παραμόρφωσης:

$$\underset{(1xn)}{[B(\xi)]}=\underset{(1x1)}{[B_1(\xi)]}\underset{(1xn)}{[B_2(\xi)]}$$

Το μητρώο στιβαρότητας:

$$\underset{(nxn)}{[K]}=\int_1\underset{(nx1)}{[B(\xi)]^T}\underset{(1x1)}{[E]}\underset{(1xn)}{[B(\xi)]}A dx=\int_{\xi_1}^{\xi_{n+p+1}}\underset{(nx1)}{[B(\xi)]^T}\underset{(1x1)}{[E]}\underset{(1xn)}{[B(\xi)]}A\det\left([J(\xi)]\right)d\xi$$

Και η αριθμητική του ολοκλήρωση:

$$\underset{(nxn)}{[K]} = \sum_{i=1}^{n_{GP\xi}} \left(\underset{(nx1)}{[B(\xi_i)]^T} \underset{(1x1)}{[E]} \underset{(1xn)}{[B(\xi_i)]} A \det\left([J(\xi_i)]\right) w_i^{GP} \right)$$

Στην περίπτωση **2D**:

$$\underset{(3x4)}{[B_1(\xi,\eta)]} = \frac{1}{\det\left(\underset{(2x2)}{[J]}\right)} \underset{(3x4)}{\begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix}}$$

$$\underset{(4x2N)}{[B_2(\xi,\eta)]} = \begin{bmatrix} R_{1,\xi} & 0 & R_{2,\xi} & 0 & \cdots & \cdots & R_{N,\xi} & 0 \\ R_{1,\eta} & 0 & R_{2,\eta} & 0 & \cdots & \cdots & R_{N,\eta} & 0 \\ 0 & R_{1,\xi} & 0 & R_{2,\xi} & \cdots & \cdots & 0 & R_{N,\xi} \\ 0 & R_{1,\eta} & 0 & R_{2,\eta} & \cdots & \cdots & 0 & R_{N,\eta} \end{bmatrix}$$

$$\underset{(3x2N)}{[B(\xi,\eta)]} = \underset{(3x4)}{[B_1(\xi,\eta)]} \underset{(4x2N)}{[B_2(\xi,\eta)]}$$

και το μητρώο στιβαρότητας:

$$\underset{(2Nx2N)}{[K]} = \int_{\xi_1}^{\xi_{n+p+1}} \int_{\eta_1}^{\eta_{m+q+1}} \underset{(2Nx3)}{[B(\xi,\eta)]^T} \underset{(3x3)}{[E]} \underset{(3x2N)}{[B(\xi,\eta)]} t \det\left([J(\xi,\eta)]\right) d\xi d\eta$$

και με αριθμητική ολοκλήρωση:

$$\underset{(2Nx2N)}{[K]} = \sum_{i=1}^{n_{GP\xi}} \sum_{j=1}^{n_{GP\eta}} \left(\underset{(2Nx3)}{\left[B(\xi_i,\eta_j)\right]^T} \underset{(3x3)}{[E]} \underset{(3x2N)}{\left[B(\xi_i,\eta_j)\right]} t \det\left(\left[J(\xi_i,\eta_j\right]\right) w_i^{GP\xi} w_j^{GP\eta} \right)$$

Για **3D** ελαστικότητα, τα μητρώα παραμορφώσεως είναι:

$$\underset{(6x9)}{\left[B_{1(\xi,\eta,\zeta)}\right]} = \underset{(6x9)}{\begin{bmatrix} J_{11}^* & J_{12}^* & J_{13}^* & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{21}^* & J_{22}^* & J_{23}^* & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & J_{31}^* & J_{32}^* & J_{33}^* \\ J_{21}^* & J_{22}^* & J_{23}^* & J_{11}^* & J_{12}^* & J_{13}^* & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{31}^* & J_{32}^* & J_{33}^* & J_{21}^* & J_{22}^* & J_{23}^* \\ J_{31}^* & J_{32}^* & J_{33}^* & 0 & 0 & 0 & J_{11}^* & J_{12}^* & J_{13}^* \end{bmatrix}}$$

$$[B_{2(\xi,\eta,\zeta)}]_{(9x3n)} = \begin{bmatrix} R_{1,\xi} & 0 & 0 & R_{2,\xi} & 0 & 0 & \cdots & R_{n,\xi} & 0 & 0 \\ R_{1,\eta} & 0 & 0 & R_{2,\eta} & 0 & 0 & \cdots & R_{n,\eta} & 0 & 0 \\ R_{1,\zeta} & 0 & 0 & R_{2,\zeta} & 0 & 0 & \cdots & R_{n,\zeta} & 0 & 0 \\ 0 & R_{1,\xi} & 0 & 0 & R_{2,\xi} & 0 & \cdots & 0 & R_{n,\xi} & 0 \\ 0 & R_{1,\eta} & 0 & 0 & R_{2,\eta} & 0 & \cdots & 0 & R_{n,\eta} & 0 \\ 0 & R_{1,\zeta} & 0 & 0 & R_{2,\zeta} & 0 & \cdots & 0 & R_{n,\zeta} & 0 \\ 0 & 0 & R_{1,\xi} & 0 & 0 & R_{2,\xi} & \cdots & 0 & 0 & R_{n,\xi} \\ 0 & 0 & R_{1,\eta} & 0 & 0 & R_{2,\eta} & \cdots & 0 & 0 & R_{n,\eta} \\ 0 & 0 & R_{1,\zeta} & 0 & 0 & R_{2,\zeta} & \cdots & 0 & 0 & R_{n,\zeta} \end{bmatrix}$$

και

$$\underset{(6x3N)}{[B(\xi,\eta,\zeta)]} = \underset{(6x9)}{[B_1(\xi,\eta,\zeta)]}\underset{(9x3N)}{[B_2(\xi,\eta,\zeta)]}$$

Το μητρώο στιβαρότητας προσδιορίζεται ως:

$$\underset{(3Nx3N)}{[K]} = \int_{\xi_1}^{\xi_{n+p+1}}\int_{\eta_1}^{\eta_{m+q+1}}\int_{\zeta_1}^{\zeta l+r+1} \underset{(3Nx6)}{[B(\xi,\eta,\zeta)]^T}\underset{(6x6)}{[E]}\underset{(6x3N)}{[B(\xi,\eta,\zeta)]}\det\left([J(\xi,\eta,\zeta)]\right)d\xi d\eta d\zeta$$

και με αριθμητική ολοκλήρωση:

$$\underset{(3Nx3N)}{[K]} = \sum_{i=1}^{n_{GP\xi}}\sum_{j=1}^{n_{GP\eta}}\sum_{k=1}^{n_{GP\zeta}}\left(\underset{(3Nx6)}{\left[B(\xi_i,\eta_j,\zeta_k)\right]^T}\underset{(6x6)}{[E]}\underset{(6x3N)}{\left[B(\xi_i,\eta_j,\zeta_k)\right]}\det\left(\left[J(\xi_i,\eta_j,\zeta_k\right]\right)w_i^{GP\xi}w_j^{GP\eta}w_k^{GP\zeta}\right)$$

ΣΥΝΟΡΙΑΚΕΣ ΣΥΝΘΗΚΕΣ-ΕΞΩΤΕΡΙΚΑ ΦΟΡΤΙΑ

Δεν είναι εύκολο να δοθεί η φυσική σημασία άσκησης φορτίων στα control points, γιατί κάθε control point γενικά δεν είναι σημείο της γεωμετρίας και φόρτιση σε αυτό αντιστοιχεί σε φόρτιση σε μια ολόκληρη περιοχή του φορέα. Όταν όμως επιβάλλονται φορτία σε control points του συνόρου εφαρμόζονται πράγματι στο ίδιο σημείο του φορέα λόγω των παρεμβολικών συναρτήσεων σχήματος. Σε διαφορετική περίπτωση, όταν είναι κατανεμημένα πάνω στον φορέα, υπολογίζονται οι ισοδύναμες δράσεις που ασκούνται στα control points:

$$\underset{(Nx3)}{\{F\}} = \int_V \underset{(Nx1)}{\{R(x,y,z)\}}\underset{(1x3)}{f(x,y,z)}\,dV = \int_{\xi_1}^{\xi_{n+p+1}}\int_{\eta_1}^{\eta_{m+q+1}}\int_{\zeta_1}^{\zeta_{l+r+1}} \underset{(Nx1)}{\{R(\xi,\eta,\zeta)\}}\underset{(1x3)}{f(\xi,\eta,\zeta)}\det[J]\,d\zeta d\eta d\xi$$

Οι συνοριακές συνθήκες εμφανίζουν αντίστοιχη δυσκολία. Δεσμεύοντας ένα control point δεσμεύεται μερικώς η περιοχή του φορέα που επηρεάζει.

Στη συνέχεια, αφού έχει μορφωθεί το μητρώο στιβαρότητας, έχουν επιβληθεί εξωτερικές φορτίσεις και συνοριακές συνθήκες, μπορεί να λυθεί η εξίσωση και να υπολογιστούν οι άγνωστες μετακινήσεις:

$$\{L_f\} = [K_{ff}]\{D_f\} \Rightarrow \{D_f\} = [K_{ff}]^{-1}\{L_f\}$$

Με γνωστές τις μετακινήσεις μπορούν να προσδιορισθούν τα πεδία μετακινήσεων, τάσεων και παραμορφώσεων σε όλο το φορέα:

Μετακινήσεις:

$$d(\xi,\eta,\zeta) = \{R(\xi,\eta,\zeta)\}^T \{D\}$$

Παραμορφώσεις:

$$\{\varepsilon(\xi,\eta,\zeta)\} = [B(\xi,\eta,\zeta)]\{D\}$$

Τάσεις:

$$\{\sigma\} = [E][B(\xi,\eta,\zeta)]\{D\}$$

ΕΠΑΝΑΔΙΑΚΡΙΤΟΠΟΙΗΣΗ

Για την βελτίωση της λύσης που προέκυψε από την ανάλυση μπορούν να χρησιμοποιηθούν τεχνικές επαναδιακριτοποίησης. Υπάρχουν πέντε είδη επαναδιακριτοποίησης:

1. h-refinement
2. reverse h-refinement
3. p-refinement
4. reverse p-refinement
5. k-refinement

Όλες οι τεχνικές επαναδιακριτοποίησης βασίζονται στη λογική ότι πριν και μετά τη πύκνωση η γεωμετρία παραμένει η ίδια. Σε κάθε περίπτωση βρίσκονται οι νέες καρτεσιανές συντεταγμένες των control point ως

$$\underset{(mx3)}{\{P^{Fine}\}} = \underset{(mxn)}{[T^{Fine-Initial}]}\underset{(nx3)}{\{P^{Initial}\}}$$

και επαναλαμβάνεται η διαδικασία μόρφωσης του μητρώου στιβαρότητας, επιβολής εξωτερικών συνθηκών και επίλυσης.

Αναλυτικότερα:

h-refinement είναι η εισαγωγή κόμβου (knot) που συνεπάγεται και εισαγωγή control point, υπάρχει σαν τεχνική και στα FEA. Εισάγονται knot values στο knot value vector και λαμβάνονται νέες συναρτήσεις σχήματος και control points.

Το **reverse h-refinement** είναι η αφαίρεση κόμβων. Δεν είναι πάντα δυνατό να αφαιρεθούν κόμβοι, για το λόγο αυτό ο αλγόριθμος για reverse h-refinement πρέπει αρχικά να ελέγξει ότι οι κόμβοι που επιλέχθηκαν προς αφαίρεση πράγματι αφαιρούνται και στην συνέχει να υπολογίσει τις νέες θέσεις των control points.

Το **p-refinement ή order elevation**, εμπλουτίζει τη βάση των συναρτήσεων σχήματος με την αύξηση του πολυωνυμικού βαθμού. Η διαδικασία περιλαμβάνει την εισαγωγή κόμβων (h-refinement) για τη μετατροπή σε ένα σύνολο από Bezier curves, στη συνέχεια την αύξηση του βαθμού των πολυωνύμων Bezier και κατόπιν αφαίρεση των περιττών knot values για μετατροπή ξανά σε B-Spline. Η τελική πολλαπλότητα των ενδιάμεσων κόμβων αυξάνεται όσο και ο βαθμός των πολυωνύμων ώστε να διατηρηθεί η συνέχεια στους κόμβους.

Το **reverse p-refinement ή order reduction**, μειώνει το βαθμό των πολυωνύμων. Η διαδικασία είναι αντίστοιχη με το p-refinement. Αποδόμηση των B-Splines σε Bezier, μείωση του βαθμού του πολυωνύμου και ξανά αφαίρεση των περιττών knot values ώστε να μετατραπεί ξανά σε B-Spline. Ομοίως με την αντίστοιχη τεχνική reverse h-refinement, γίνεται έλεγχος αν πραγματικά μπορεί να μειωθεί ο βαθμός χωρίς να χαθεί η ακρίβεια στη γεωμετρία. Αν γίνεται, τότε μειώνεται και η τελική πολλαπλότητα των κόμβων στο knot value vector ώστε η συνέχεια να παραμείνει η ίδια.

Τέλος, το **k-refinement** είναι μια νέα τεχνική μοναδική στην ισογεωμετρική ανάλυση. Εφαρμόζεται αρχικά ένα p-refinement για την αύξηση του βαθμού των πολυωνύμων και στη συνέχεια ένα h-refinement το οποίο εισάγει κόμβους με υψηλή συνέχεια. Ο λόγος της υψηλής συνέχειας, είναι ότι οι κόμβοι εισήχθησαν μετά το p-refinement και έτσι δεν αυξήθηκε η πολλαπλότητά τους.

ΠΡΟΣΑΡΜΟΣΤΙΚΗ ΙΕΡΑΡΧΙΚΗ ΕΠΑΝΑΔΙΑΚΡΙΤΟΠΟΙΗΣΗ

Πέρα όμως από τις πέντε κλασσικές εκδοχές επαναδιακριτοποίησης που προαναφέρθηκαν υπάρχουν κι άλλες τοπικού και προσαρμοστικού χαρακτήρα. Έχουν γίνει προσπάθειες ώστε να υπάρχει δυνατότητα τοπικής πύκνωσης του δικτύου που συνεπάγεται εισαγωγή μειωμένου αριθμού control points. Η προσπάθεια αυτή δημοσιοποιήθηκε αρχικά απο την A.Vuong το 2011. Στην πρόταση της η A.Vuong υιοθετεί τις αρχές της τεχνολογίας CAD για την διακριτοποίηση της γεωμετρίας προσαρμοσμένη στις ανάγκες της ισογεωμετρικής ανάλυσης. Επιχειρεί επιτυχώς όπως αποδεικνύεται να αντικαταστήσει τοπικά συναρτήσεις σχήματος με πυκνότερες ώστε να βελτιώσει την λύση.

Η ιεραρχική πύκνωση του δικτύου λοιπόν, είναι μια τεχνική τοπικού εμπλουτισμού του υπολογιστικού πεδίου, κατά την οποία γίνεται αντικατάσαση μέρους του αδρού δικτύου με ένα πυκνότερο. Από το 2011 έως και σήμερα οι προτεινόμενες μέθοδοι είναι πολλές και βασιζόμενες στην πρωταρχική ιδέα προσπαθούν να εντοπίσουν τον βέλτιστο και πιο αποδοτικό μηχανισμό τοπικής επαναδιακριτοποίησης.

Κοινός γνώμονας όλων αυτών των μεθόδων η διατήρηση των ιδιοτήτων των βάσεων ώστε να είναι κατάλληλες για τις ανάγκες της ανάλυσης. Δύο απ' αυτές, ίσως και οι σημαντικότερες, είναι η γραμμική ανεξαρτησία των συναρτήσεων βάσεως και και η επιτακτικότητα το άθροισμα των τιμών τους σε κάθε σημείο της βάσης να ισούται με 1. Η πρώτη ιδιότητα εξασφαλίζεται με χρήση της θεωρίας του Kraft που πραγματεύεται την πύκνωση γεωμετρικών δικτύων. Στην δεύτερη ιδιότητα έγκειται και η δυσκολία επιλογής σωστής μεθόδου τοπικής ιεραρχικής επαναδιακριτοποίησης. Οι περισσότερες μέθοδοι εισάγουν κάποια βάρη τα οποία πολλαπλασιάζουν τις συναρτήσεις σχήματος ώστε να ικανοποιείται η τελευταία ιδιότητα.

Στην παρούσα διπλωματική εξετάστηκαν και και αξιολογήθηκαν σε θεωρητικό και ποιοτικό επίπεδο κάποιες από αυτές τις μεθόδους. Στόχος ήταν η σύγκριση των χαρακτηριστικών τους. Παρουσιάζονται λοιπόν οι προτάσεις της Gianneli (truncated B-Splines), των Bornemann και Cirak (HB-Splines) και του D. Shillinger που αποτελεί ένα υβρίδιο των προηγούμενων δύο μεθόδων. Για τις περισσότερες μεθόδους, βασική προυπόθεση αποτελεί η δημιουργία της κλασσικής ιεραρχικής βάσης. Στην συνέχεια συνήθως ακολουθεί μια διαδικασία αναγνώρισης των συναρτήσεων που την αποτελούν, καθώς αυτές προέρχονται από πολλά συνεχόμενα (συνήθως) επίπεδα πύκνωσης. ΄Επειτα, πρέπει να αποτυπωθούν με ακρίβεια οι αλληλεπιδράσεις αυτών των συναρτήσεων και να αποθηκευτούν με τρόπο οργανωμένο και εύκολα επεξεργάσιμο.

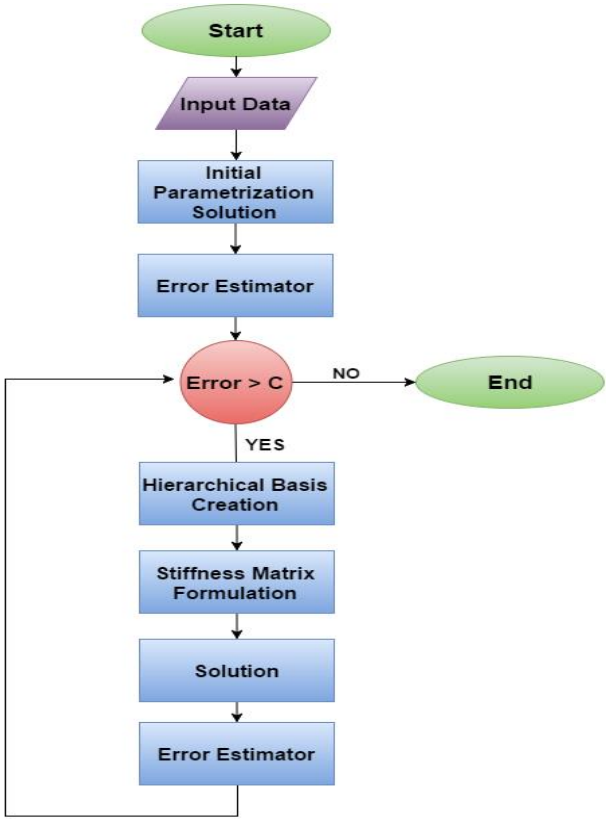
Στις περισσότερες από αυτές τις μεθόδους ως βάρη για την ικανοποίηση της ιδιότητας του partition of unity ($\Sigma N_i=1$ σε κάθε θέση) χρησιμοποιούνται εκείνα που ικανοποιούν την ιδιότητα της υποδιαίρεσης (subdivision property). Σύμφωνα με αυτή την ιδιότητα κάθε συνάρτηση σχήματος μπορεί να γραφτεί ως γραμμικός συνδυασμός πυκνότερων συναρτήσεων σχήματος. Όλες λοιπόν οι στρατηγικές που παρουσιάζονται επικεντρώνονται στην αντικατάσταση αδρών συναρτήσεων σχήματος με πυκνότερες που μπορούν να τις εκφράσουν. Η ιδιότητα της γραμμικής ανεξαρτησίας ικανοποείται με την αφαίρεση όλων των αδρών συναρτήσεων σχήματος που μπορούν να εκφραστούν πλήρως απο τις πυκνότερες που έχουν εισαχθεί. Η Giannelli προτείνει και την αφαίρεση τμημάτων συναρτήσεων βάσεων που μπορούν να εκφραστούν μερικώς από τις πυκνότερες για την περαιτέρω μείωση των αλληλεπιδράσεων των συναρτήσεων σχήματος που επιβαρύνουν το μητρώο δυσκαμψίας με επιπλέον στοιχεία. Παράγονται δηλαδή μητρώα με λιγότερους μη-μηδενικούς όρους, μειωμένο bandwidth, αλλά και καλύτερο conditioning number. Ο D. Shillinger με την σειρά του συνδυάζοντας τις δύο προηγούμενες στρατηγικές επιχειρεί να διατηρήσει τον τοπικό χαρακτήρα της ιεραρχικής διακριτοποίησης εισάγοντας τις λιγότερες δυνατές πυκνότερες συναρτήσεις σχήματος.

Όλες όμως οι παραπάνω μεθοδολογίες εστιάζουν μόνο στην μόρφωση των βάσεων των συναρτήσεων σχήματος. Οδηγούν δηλαδή στην μόρφωση του εμπλουτισμένου μητρώου δυσκαμψίας. Στην περίπτωση προσαρμοστικής επαναδιακριτοποίησης η επίλυση του μητρώου στιβαρότητας που προκύπτει από κάθε επίπεδο τοπικής πύκνωσης επαναλαμβάνεται εξ αρχής. Για αυτόν ακριβώς τον λόγο ο Wu και οι συνεργάτες του, πρότειναν μια μεθοδολογία κατα την οποία πραγματοποείται επίλυση μικρότερων

μητρώων στιβαρότητας τα οποία ανταποκρίνονται σε κάθε επίπεδο της ιεραρχίας. Μ' αυτό το τρόπο οι βαθμοί ελευθερίας που έχουν αποδεκτό σφάλμα επιλυόνται μόνο μία φορά και στη συνέχεια η επιρροή τους στον φορέα μεταφέρεται ως συνοριακές συνθήκες στους βαθμούς ελευθερίας που αντιστοιχούν στο εισαγόμενο πυκνότερο επίπεδο. Η μεθοδολογία αυτή οδηγεί πρακτικά σε κατακερματισμό του καθολικού μητρώου στιβαρότητας σε μικρότερα μητρώα στιβαρότητας τα οποία μπορούν να επιλυθούν ανεξάρτητα μεν, με την σωστή αλληλουχία δε.

Σε αυτό το σημείο αξίζει να διευκρινιστεί ο όρος 'ιεραρχική' επαναδιακριτοποίηση που χρησιμοποίθηκε έως τώρα. Ο όρος ιεραρχική αναφέρεται αποκλειστικά και μόνο στην ιεραρχική σχέση που διέπει τις συναρτήσεις σχήματος και των σχηματισμό των εμπλουτισμένων βάσεων. Δεν αναφέρεται στον ιεραρχικό σχηματισμό του μητρώου στιβαρότητας. Η ιεραρχική διατύπωση του μητρώου θα ήταν κάτι εξαιρετικά χρήσιμο στην τοπική πύκνωση του δικτύου καθώς δεν απαιτεί την εξαρχής μόρφωση του σε κάθε επίπεδο διακριτοποίησης. Σε μια τέτοια περίπτωση και σε συνδυασμό με την μέθοδο του Wu η ιεραρχική επαναδιακριτοποίηση θα ήταν υπολογιστικά ιδιαίτερα οικονομική. Η παράμετρος αυτή όμως είναι κάτι στο οποίο δεν έχει υπάρξει ακόμα κάποια ερευνητική εξέλιξη.

Με γνώμονα όλα όσα προαναφέρθηκαν προτείνεται αλγόριθμος προσαρμοστικής ιεραρχικής επαναδιακριτοποίησης για την επίλυση δισδιάστατων προβλημάτων επίπεδης έντασης και επίπεδης παραμόρφωσης. Η λογική του προτεινόμενου αλγόριθμου συνοψίζεται στο παρακάτω διάγραμμα ροής:



Η επαναδιακριτοποίηση πραγματοποιείται με την επιλογή των στοιχείων με το υψηλότερο σφάλμα. Εφόσον προσδιορισθούν τα στοιχεία για επαναδιακριτοποίηση, αντικαθιστώνται με πυκνότερες συναρτήσεις σχήματος όλες οι συναρτήσεις σχήματος που διέρχονται από τα στοιχεία αυτά. Οι συναρτήσεις που εισάγονται πολλαπλασιάζονται με κατάλληλους συντελεστές ώστε το άθροισμα των τιμών των συναρτήσεων σχήματος της ιεραρχικής βάσης να είναι πάντα 1. Οι συνετελεστές αυτοί προσδιορίζονται από την ιδιότητα της υποδιαίρεσης (subdivision property).

$$N_{i,\mathrm{p}}^{\ell} = \sum_{k=0}^{p+1} S_{i,k}^{p} N_{2i+k,p}^{\ell+1}(\xi)$$

όπου

$$S_{i,k}^{p} = \frac{1}{2^p}\left(\frac{p+1}{k}\right) = \frac{1}{2^p}\frac{(\mathrm{p+1})!}{(\mathrm{p+1}-k)!\mathrm{k}!}$$

Προτείνονται επίσης κατάλληλοι συντελεστές για συναρτήσεις μειωμένης συνέχειας.

Εξασφαλίζεται επίσης η γραμμική ανεξαρτησία των συναρτήσεων σχήματος αφαιρόντας όλες τις συναρτήσεις που μπορούν να περιγραφούν πλήρως από τις πυκνότερες συναρτήσεις που έχουν εισαχθεί.

Όσον αφορά την εισαγωγή των νέων συναρτήσεων σχήματος είναι ιδιαίτερα σημαντικό να βρεθούν οι θέσεις των νέων control points ώστε η γεωμετρία να παραμείνει η ίδια. Αποδεικνύεται ότι :

$$\left\{u^{l+1}\right\} = [S]\cdot\left\{u^{l}\right\}$$

Όπου [S] μητρώο μετασχηματισμού το οποίο περιέχει τους συντελεστές που προαναφέρθηκαν και τις σχέσεις μεταξύ των συναρτήσεων δύο διαδοχικών ιεραρχικών επιπέδων. Τονίζεται πως τελικώς οι συντελεστές πολλαπλασιάζουν τις συντεταγμένες των control points κι όχι τις ίδιες τις συναρτήσεις σχήματος.

Με δεδομένα όλα τα παραπάνω παρουσιάζεται η ιδιαίτερη σημασία της εκτιμήτριας σφάλματος που αποτελεί το κλειδί για την υλοποίηση ενός αλγόριθμου προσαρμοστικής επαναδιακριτοποίησης. Στην παρούσα διπλωματική προτείνεται μια απλή εκτιμήτρια σφάλματος βασισμένη στις κλίσεις του πεδίου των μετατοπίσεων σε κάθε στοιχείο. Η εκτίμηση σφάλματος στοιχείου δίνεται από την παρακάτω σχέση

$$\varepsilon_e = \frac{1}{V_e}\left(\int_{\Omega_e} |\nabla u|^2 \, d\Omega_e\right)^{1/2}$$

που αποδεικνύεται πως ισούται με

$$\varepsilon_e = \frac{1}{\det[J]td\xi d\eta}\left(\int\int \{d\}^T [B_2]^T \left[\bar{J}^{-1}\right]^T \left[\bar{J}^{-1}\right][B_2]\{d\}\det[J]d\xi d\eta\right)^{1/2}$$

όπου

$$\left[\bar{J}^{-1}\right]=\frac{1}{\det\left[J\right]}\begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ -J_{21} & J_{11} & 0 & 0 \\ 0 & 0 & J_{22} & -J_{12} \\ 0 & 0 & -J_{21} & J_{11} \end{bmatrix}$$

Και ολοκληρώνεται αριθμητικά ως

$$\varepsilon_e=\frac{1}{\det\left[J\right]t d\xi d\eta}\sum_{i=1}^{GP\xi}\sum_{j=1}^{GP\eta}\{d\}^T\left[B_2\right]^T\left[\bar{J}^{-1}\right]^T\left[\bar{J}^{-1}\right]\left[B_2\right]\{d\}\det\left[J\right]w_i^{GP\xi}\cdot w_j^{GP\eta}$$

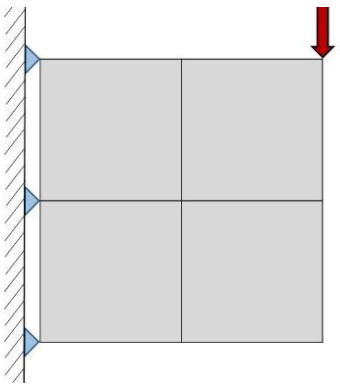
Εάν ένα στοιχείο έχει σφάλμα το οποίο ξεπερνά ως ποσοστό σφάλματος επί του συνολικού την σταθερά C(%) τότε υπόκειται σε πύκνωση. Συνοπτικά αυτό συμβαίνει όταν

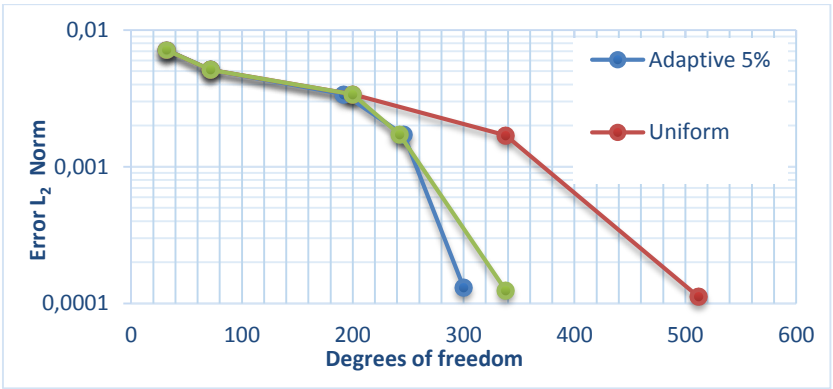
$$\varepsilon_e > C\sum_{1}^{n_e}\varepsilon_e$$

Τέλος παρουσιάζονται δύο εφαρμογές επίπεδης έντασης για την αξιολόγηση της προσαρμοστικής ιεραρχικής επαναδιακριτοποίησης σε σχέση με την κλασική μέθοδο του h-Refinement.

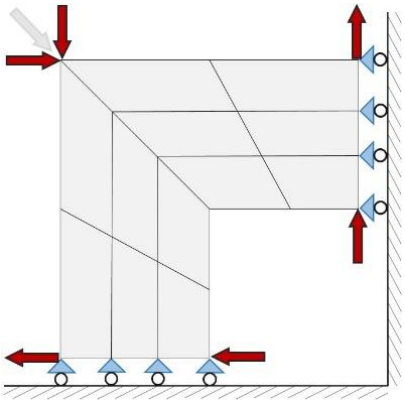
Τα αποτελέσματα αυτών των εφαρμογών συνοψίζονται στα παρακάτω διαγράμματα.

ΠΡΟΒΟΛΟΣ 2D





L-SHAPE 2D



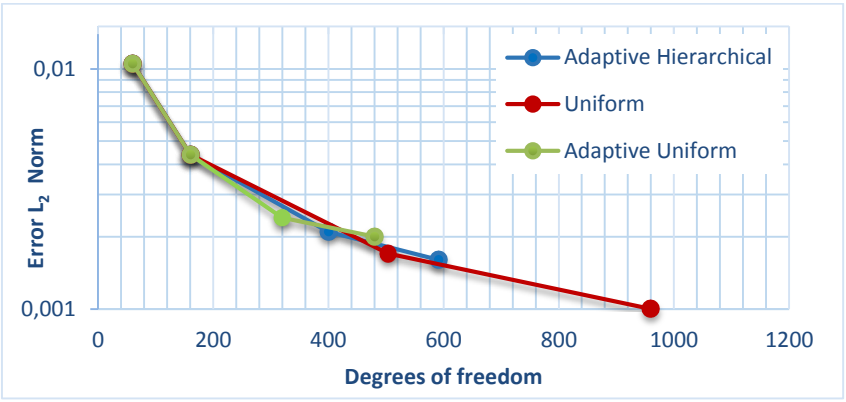

0,01
0,001
Error L_2 Norm
0
200
400
600
800
1000
1200
Degrees of freedom
Adaptive Hierarchical
Uniform
Adaptive Uniform

TABLE OF CONTENTS

1. INTRODUCTION

1.1 FINITE ELEMENT METHOD

1.1.1 Basic Idea

The basic idea behind the Finite Element Method is to solve a complex problem of partial differential equations by converting it into a system of simplified algebraic equations. In order to accomplish that, FEA computes the solution in discrete and finite points (nodes) defined over the examined structure. Now, the field of displacements can be approximately calculated for any internal point of the structure. It is straightforward that due to the classical equation of elasticity, strains and stresses can be calculated as well. FEA is able to provide with reliable results, even though it is an approximate and numerical method, a feature that made it very attractive over the past decades for solving engineering problems such as heat transfer, static or dynamic analysis of structures, solid-fluid interaction, wave propagation and even fluid dynamics.

FEA approximates the solution with piecewise polynomial functions, called shape functions N, which calculate the displacement value {U} at any internal point (x,y,z) of the element by interpolation of the nodal displacements {d}, where n_e is the number of the discretized finite elements over the structure.

$$\underset{(3\times1)}{\{U(X,Y,Z)\}} = \underset{(3\times3n_e)}{[N(X,Y,Z)]} \cdot \underset{(3n_e\times1)}{\{d\}} \tag{1.1}$$

It is easy to notice that as the number of elements increases, such are the degrees of freedom and the accuracy of the approximate solution. Along with the aforementioned aspects element number increase leads to significant computing cost. It is easily comprehended that computing cost is not crucial for simple problems with a few degrees of freedom but for complex and large scale problems.

The problem is directly downsized from infinite unknowns to a finite number of degrees of freedom. The next step is to define stress and strain matrices and their interconnection. In the case of linear elastic problems these interconnections are defined by Hooke's constitutive law.

$$\underset{(6x1)}{\{\sigma\}} = \begin{bmatrix}\sigma_X & \sigma_Y & \sigma_Z & \sigma_{XY} & \sigma_{YZ} & \sigma_{ZX}\end{bmatrix}^T \text{ (3D case)} \tag{1.2}$$

$$\underset{(6x1)}{\{\varepsilon\}} = \begin{bmatrix}\varepsilon_X & \varepsilon_Y & \varepsilon_Z & \gamma_{XY} & \gamma_{YZ} & \gamma_{ZX}\end{bmatrix}^T \text{ (3D case)} \tag{1.3}$$

$$\underset{(6x1)}{\{\sigma\}} = \underset{(6x6)}{[E]} \cdot \underset{(6x1)}{\{\varepsilon\}} \tag{1.4}$$

Deformation matrix [B] evaluates strains from nodal displacements in any material point of the model with coordinates (x,y,z).

$$\underset{(6x3n_e)}{\left[B(x,y,z)\right]} = \underset{(6x9)}{\left[B_1(x,y,z)\right]} \cdot \underset{(9x3n_e)}{\left[B_2(x,y,z)\right]} \tag{1.5}$$

$$\underset{(6x1)}{\left\{\varepsilon(x,y,z)\right\}} = \underset{(6x3n_e)}{\left[B(x,y,z)\right]} \cdot \underset{(3n_e x1)}{\left\{D\right\}} \tag{1.6}$$

where n_e is the number of nodes per element.

The local stiffness matrix [k] of each model's element is given by the following integral.

$$\underset{(3n_e x3n_e)}{\left[k\right]} = \int_V \underset{(3n_e x6)}{\left[B(x,y,z)\right]^T} \cdot \underset{(6x6)}{\left[E\right]} \cdot \underset{(6x3n_e)}{\left[B(x,y,z)\right]} dV \tag{1.7}$$

Distributed loads can be transformed into equivalent nodal loads, according to the next equation.

$$\underset{(3n_e x1)}{\left\{r\right\}} = \int_V \underset{(3n_e x3)}{\left[N(x,y,z)\right]^T} \cdot \underset{(3x1)}{\left\{f(x,y,z)\right\}} dV \tag{1.8}$$

The local stiffness matrix [k] of each element is added to the total stiffness matrix [K] of the structure. The displacement vector is calculated with Gauss elimination by the system:

$$\underset{(3nx1)}{\left\{R\right\}} = \underset{(3nx3n)}{\left[K\right]} \cdot \underset{(3nx1)}{\left\{D\right\}} \tag{1.9}$$

where n is the model's total node number.

We observe that shape functions [N] and deformation matrix [B] depend on the Cartesian coordinates (x,y,z). This leads to iteration problems when complex geometries (e.g. circles or conic sections in general) are involved. The solution, which contributed to the further generalization of FEM, along with a more efficient iteration algorithm, is isoparametric elements. The basic idea is the existence of a parent element in the parameter space, which can be modeled as a regular shape (e.g. a cube, square or an equilateral triangle). Each element in the physical space (the "real" modeling space) can be described using a linear combination of the parent element's shape functions. Hence, geometry can be approximated by the same functions used for the solution field. This explains the name "isoparametric".

1.1.2 Method Development

A variety of complex problems were until recently considered unsolved considering the large computational effort needed. This fact made their analysis practically impossible. The first application of finite element method was the triangulation of airplane wings, in order to determine how they are overworked by wind forces. It was the requirement of numerous hand calculations that made the process inapplicable. The progress of computer technology and computational methods (computational mechanics & geometry) enabled the evolution and programming of finite element method to tackle these obstacles by solving complicated problems in much less time and with far less human effort.

FEA is nowadays so widely spread and accepted by the scientific community for the following reasons. Firstly FEA can be used in problems that require numerical approximation, as analytical solution is not attainable. In addition, the enforcement of complex geometries, physical loads and material properties to the simulated model is possible, without arising any significant difficulties. It also enables the connection of complex composite materials, while it can be easily refined and altered to approach the accuracy requirements of each problem. Finally, the introduction of dynamic analysis and non-linear material and geometry behavior enables the approximation of real materials and their actual behavior in numerous cases of physical problems.

The need to apply this innovative technique in industrial scale encouraged the development of finite element software. These programs needed to adopt a double nature by supporting both design and analysis process, which is a very difficult task. The combination of graphical representation and numerical analysis with finite elements would simplify engineering tasks, as it would focus only on the essential aspects of each problem, neglecting the need of complex calculations.

Nowadays, commercial software partially incorporates both graphic design and finite element analysis. State of the art programs are now able to solve a wide range of problems from soil mechanics and structural analysis to thermodynamics and solid fluid interaction, covering practically most problems an engineer can encounter. The additional combination of non–linear behavior and dynamic analysis transforms FEA programs into the proper tool for every engineer to face sufficiently even the most demanding and cost intensive problems.

Nowadays the method of finite elements is not only used by engineers. All scientific fields have become acquainted to this method as it became a very handful tool. For instance, doctors with aid of engineers analyze blood flow in vessels to determine weak points of the walls and diagnose and prevent heart diseases (thoracic and abdominal aortic aneurysm). At the same time, all mechanical tools and substitutes used in surgeries to replace human parts could be designed via FEA. In general, FEA tends to be the widest utilized tool for all fields of science.

1.1.3 Drawbacks

Even though the evolution of finite element method has been rapid since its birth, main drawbacks have not been overcome yet. For example, exact geometrical representation of the model is almost impossible. Even isoparametric elements can only produce a more precise but still approximate mesh of the geometry. The most challenging tasks, that engineers face today, often require exact geometrical representation in order to achieve the desired accuracy. Except from mesh generation of CAD design, the initial geometry plays no role in the analysis. This seems to be wrong, as the accurate geometry is replaced with an approximate one. As expected, it produces a vast number of issues. The inevitable geometrical approximation means there will be convergence errors by definition, regardless of the solution methods and the available computational power. This affects the efficiency of the solution.

In case a better approximation is required, refinement algorithms will return to the initial geometry and produce a new and more precise one from the beginning. The new, fine mesh cannot be directly produced from the previous. This means that procedures already completed have to be repeated in order to create the new mesh. Precious analysis time is required and the geometrical differences between the coarse and fine mesh make it difficult to compare the result. It is obvious that a new approach is needed that will integrate CAD programs and FEA software. Unfortunately, computational geometry and numerical analysis have been developed in different eras. Computer aided design has evolved greatly since its birth. On the other hand, finite element design geometries are unable to follow the rapid CAD evolution. These problems were always present throughout the history of FEM. Complicated geometries used in mechanics led to the development of algorithms and computational methods to minimize this effect. The problem is that the definition of finite elements does not facilitate the progress towards the full cooperation of design and analysis.

The solution to the drawback of geometric approach mentioned above is to embrace the development of Isogeometric Analysis (IGA) using NURBS, T-Splines and subdivision surfaces, taking advantage of the method's main idea, which suggests that any shaped geometry remains intact and the analysis is performed on the exact geometric model. Shape functions that are used in isogeometric analysis are in fact those used for design. This breakthrough of IGA allows the analysis of structures with complex geometry, without making approximations. In addition, a designer is able to draft any object using less control points (equivalent to the nodes of FEA). When the object is about to undergo analysis, the engineer can effortless obtain the suitable mesh. Even when refinement procedure is needed, the engineer does not have to go back to the initial geometry. Refinement is performed always directly on the imported geometry, following the rules of design. Consequently, geometry does not change, leading to important time economy by procedures that will not be performed again.

1.2 COMPUTER AIDED DESIGN

1.2.1 Historical Overview

Computer Aided Design (CAD) emerged in the 1950s from the automotive, shipyard and aircraft industries. In those times, designers were able to produce accurate drafts by hand, but when ship cross sections had to be drafted in real-life size, pencils could not help anymore and things became a bit more complex. The main problem was the definition of a real-size curve, which smoothly interpolated several predetermined points in order to create the shell of the ship. This task was usually carried out in the loft of a building, due to the large amount of space needed. The loftsman, as he was called, used easy-to-bend pieces of steel or wood, called splines, in order to interpolate the points. In order to maintain the spline's shape, he usually put weights on them on certain points.

The development of NURBS arose from the need to effectively represent freeform surfaces. Two engineers in France stood at the forefront of this approach, Pierre Bezier from Renault and Paul de Casteljau from Citroen. Bezier's work was the first to reach publication, and soon after the CAD industry started using and enhancing Bezier curves. However, certain disadvantages of Bezier curves led to the search for more convenient forms of representation. Researchers finally introduced B-Splines, which were similar to Bezier curves, meaning that the curve was defined by a set of points, called the "control points", but their number was independent of the polynomial order of the curve. B-Splines were a generalization of Bezier curves, but they were also more convenient to edit. Changing a point didn't influence the whole curve, but only part of it.

DATE	SUBJECT	AUTHOR
1912	Bernstein Polynomials	Bernstein
1946	Schoenberg coins the name ''SPLine''	Schoenberg
1956	Subdivision seminal ideas	Rham
1959	De Casteljau Algorithm	De Casteljau
1966-1972	Bezier curves and surfaces	Bezier-Faget
1971-1972	Cox-de Boor recursion	Cox-de Boor
1972	B-SPLines	Riesenfeld
1974	Subdivision seminal ideas	Chaikin
1975	NURBS	Versprille
1978	Subdivision Surfaces	Catmull-Clark and Doo-Sabin
1989	Oslo knot insertion algorithm	Oslo
1987	Loop subdivision	Loop
1987-1989	Polar forms, blossoms	Ramshaw
1996-present	Triangular and tetrahedral B-SPLines	Lai-Schumaker
1997	SPLine finite elements	Sabin
2003	T-SPLines	Sederberg
2008	Replace trimmed NURBS surface with untrimmed T-SPLines	Sederberg et al.

Figure 1.1.
History of CAD technologies.
(Iakovos Antonios 2015)

Another problem is that even B-Splines cannot produce an exact representation of conic sections. This is where NURBS came along. Ken Versprille was the first to work with NURBS on his dissertation in 1975. Later, when he acquired a top position in Computervision, the company began to support NURBS. Boeing, in its ambitious project to create a single curve representation that included Bezier curves and conic sections, became the first to industrialize NURBS. After that, NURBS began to spread across the CAD industry. They possessed a lot of interesting attributes. The parameterization of the whole curve was downsized to a few control points coordinates, allowing numerically stable mathematical procedures and easy modification. Cartoon characters, videogame graphics, ships, cars, airplanes were designed using NUBRS. This led to major investments from research and industrial faculties. Graphic designers became accustomed to them and students were taught about the theory and implementation of NURBS in real-life problems. This cycle's milestones presented in **Fig. 1.1** led to the increasing popularity of Non-Uniform Rational B-Splines in the CAD industry.

Today, even though many other forms of more suitable representations exist, such as T-Splines, Polycube Splines and Subdivision surfaces, NURBS still hold a large share of the market. Many platforms exist for NURBS and designers still find it easier to use them. Many handful tools have been developed over the years for them (knot insertion, order elevation, curve fitting, patching). This makes them more attractive and easy-to-use than newer techniques with less industry experience behind them.

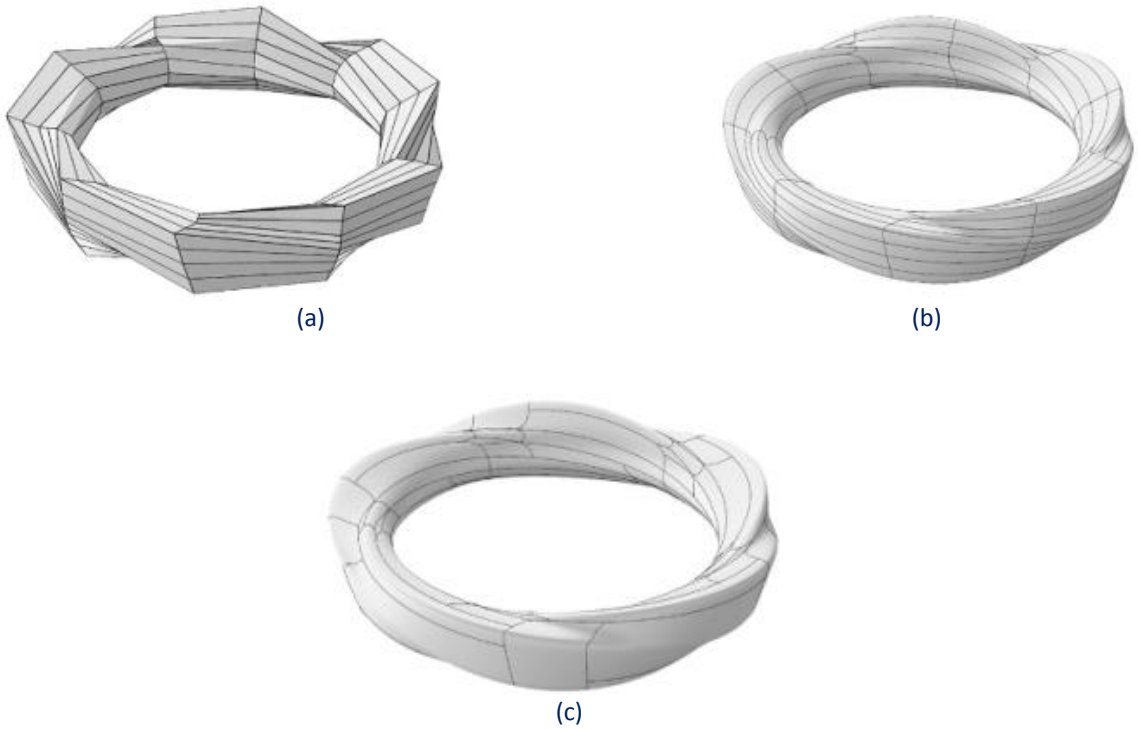


Figure 1.2.
Representing an arbitrary geometry with: **(a)** Subdivision Surfaces, **(b)** Smooth Subdivision Surfaces, **(c)** NURBS surfaces

1.3 ISOGEOMETRIC ANALYSIS

1.3.1 Introduction

Considering the aforementioned drawbacks of the classical finite element method researchers worldwide have been trying to overcome them. Their aim was to eliminate the geometry error that every analysis inherited due to the different mesh of design and analysis.

It was clear that integration between the two technologies CAD and CAE was necessary, in order to achieve better numerical results as well as to save valuable computational time in terms of CPU-time and generation of an analysis suitable mesh. Such were the reasons that led Professor Hughes and his colleagues to introduce Isogeometric Analysis (IGA) back in 2003 and make a breakthrough in computational mechanics.

At the same time, Isogeometric Analysis was the first successful systematically proposed method so far, suitable for implementation and programming. IGA has been since then a great topic for academic research for many engineers around the world as it shown enormous potential for engineering application that could someday overthrow FEA as the leading technology in engineering software.

The main idea was to use the same exact mesh that emerged from the design process in CAD software into the analysis. It is clear that this concept would eliminate completely the time needed for assembling the approximate finite element mesh.

The time for assembling the analysis mesh is approximately 70% of the total analysis time in modern applications of FEA in complicated structures. It was the need for reduced computational time that initially motivated the isogeometric analysis concept in order to merge geometry design and mesh generation. In FEA software, practically around 80% of the overall analysis time is consumed in efforts to acquire a satisfactory approximate mesh, which has to be quite close to the accurate mesh of the geometry and suitable for analysis purposes.

1.3.2 Basic Idea

The basic idea behind Isogeometric Analysis was to merge the Computer Aided Design with the Computer Aided Engineering technology. In order to achieve this goal, the same mesh for design and analysis was needed. Hughes et al used the shape functions of the design process directly into the analysis.

While CAD community uses NURBS, T-Splines and subdivision surfaces, the CAE community generally uses Lagrange polynomials to approximate the geometry. FEA's approximate mesh necessitates a re-parameterization of the CAD geometry, which makes labor cost quite intensive and introduces automatically geometrical error. In 2003, the research on isogeometric analysis started to focus on the question if finite element analysis could be done with Non-Uniform Rational B-Splines (NURBS), the most widely used computational geometry tool in commercial CAD programs.

Recent research on isogeometric analysis uses Non-Uniform Rational B-Splines (NURBS) as shape functions, because they are extremely popular in engineering design systems. It has been shown that NURBS-based finite elements are very well suited for computational analysis, leading to more accurate results in comparison with standard finite elements based on Lagrange polynomials.

NURBS were until lately the main shape functions used in isogeometric analysis. In 2008, T-Splines were introduced as a worthy "opponent" that holds all the benefits of NURBS and at the same time permits local refinement. Water tightness, less control points, more design capabilities, but also sophisticated nature and complex implementation are its main features.

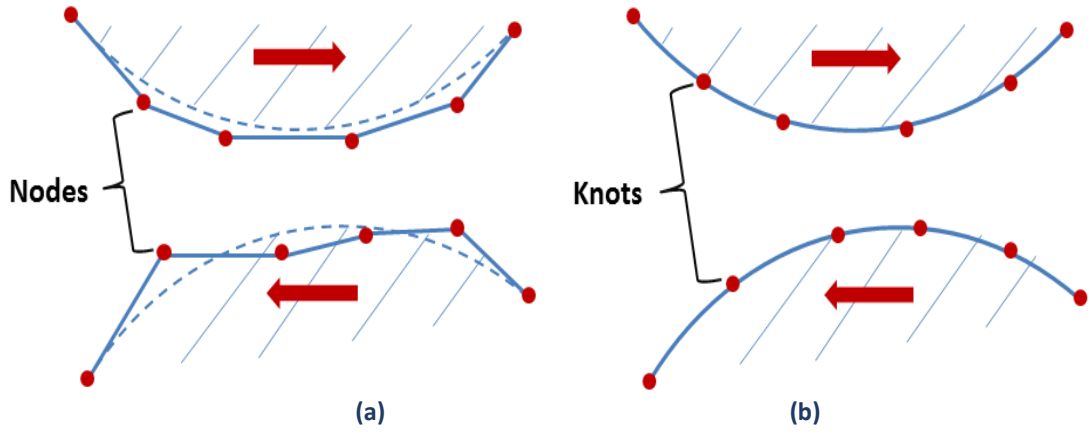


Figure 1.3.
Isogeometric Analysis is the powerful generalization of Finite Element Analysis. **(a)**Finite Element Analysis. **(b)** Isogeometric Analysis.
(Iakovos Antonios 2015)

1.3.3 Index, Parameter and Physical Space

Generally, finite element problems are not being handled in their physical form. In most occasions, the exact solution of a natural problem is neither possible nor necessary. The actual objective is to find an accurate solution that satisfies a selected convergence criterion. That means that FEA practitioners have developed methods to ease the numerical approach of each problem, by converting their true geometry into an analysis and implementation suitable one. The ultimate challenge for an engineer is to balance between accuracy and time. Design and analysis of extraordinary geometries is a powerful asset for modern engineers, who are capable of facing surprisingly more complicated problems. The assumption of new spaces that would extend beyond the physical dimensions of the problem became necessary. Accurate geometrical representations of the natural model are designed in the familiar Cartesian system, called Physical space. Additionally, it is very helpful to envision a complex structure in an imaginary, basic space, where all geometries can be represented as lines, rectangles and cuboids. This is Parameter space. This approach is far from new; it is already known from the isoparametric concept in finite element methods. The parameter space utilized in isogeometric analysis, however, holds some major differences. Furthermore, isogeometric analysis also introduces the Index space. This additional space plays an important role for some kinds of Splines, but it is only auxiliary for NURBS.

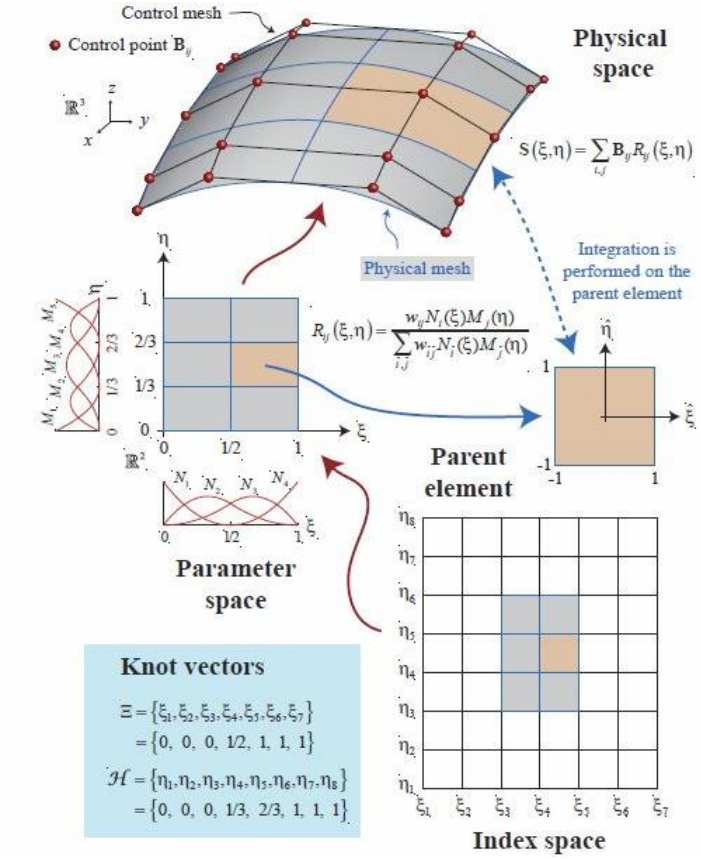


Figure 1.4.
The different mappings and spaces of Isogeometric Analysis.
(Hughes et al 2007)

1.3.3.1 Physical Space

Physical space is the already known Cartesian space, where the real model is represented. Simple orthogonal shapes from parameter space are transformed into complex entities in the physical space. Physical coordinates of the control points play a major role in the aforementioned mapping, but an equally drastic role is set upon basis functions. In fact, for a given set of control points, only a single set of basis functions can lead to the same geometry. We will examine this thoroughly later. Control points can often be seen outside the model in physical space in contrast to FEM's nodes which always belong to the mesh as depicted in **Fig. 1.3**. It is one of the reasons NURBS and Spline entities in general can accurately represent multiple types of geometries and the understanding of this peculiarity is one of the many challenges of isogeometric analysis.

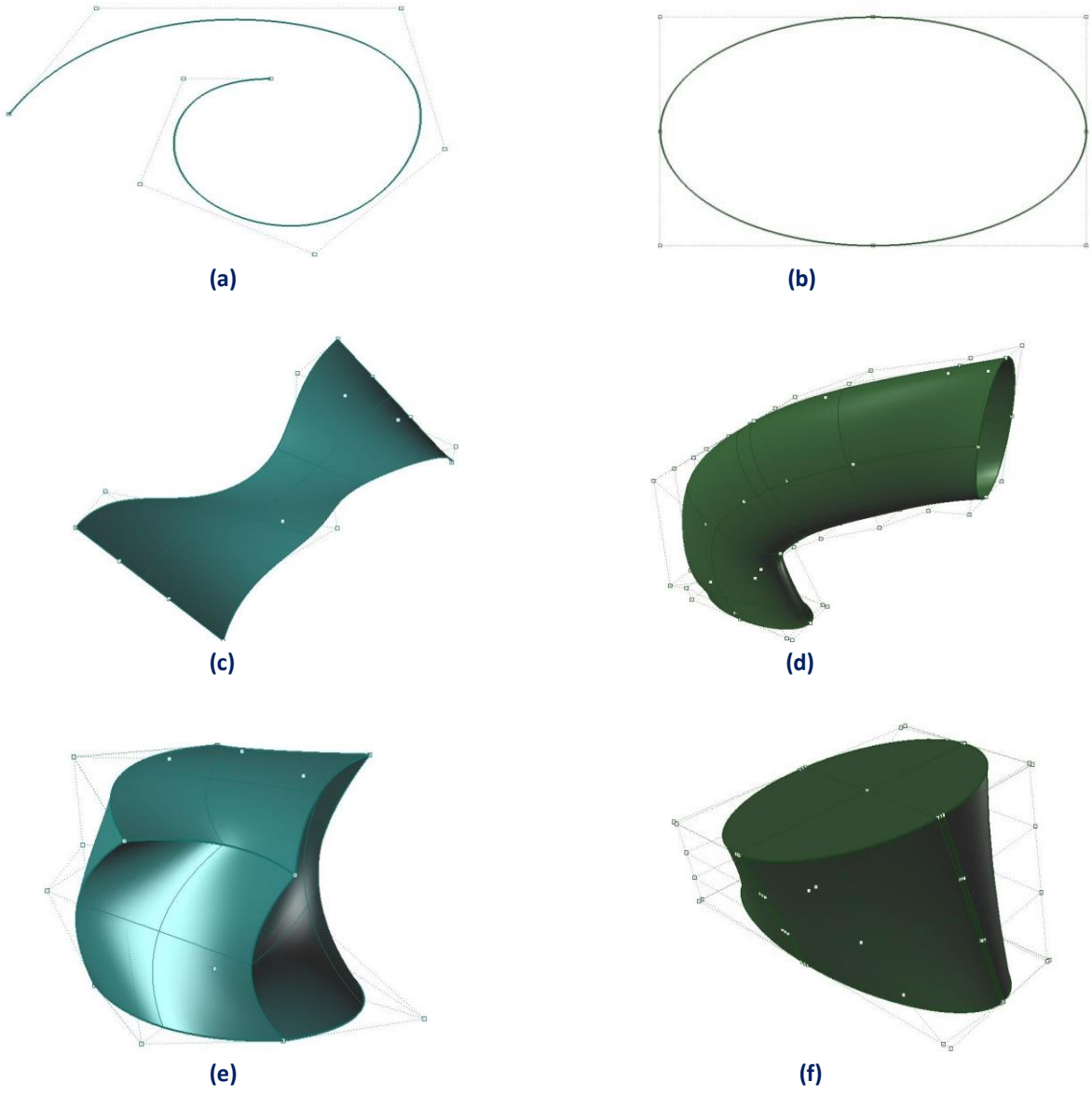


Figure 1.5.
(a) B-Spline curve, **(b)** NURBS curve, **(c)** B-Spline surface, **(d)** NURBS surface **(e)** B-Spline solid, **(f)** NURBS solid

1.3.3.2 Parameter Space

Parameter space is a representation of the model with respect to knots. Spline entities are always represented as orthogonal shapes in parameter space. Only lines, rectangles and cuboids exist here. In order to transform those simple patterns to virtually unlimited, complex geometries, the application of a mapping from parameter to physical space is required. Hence, parameter space is a primitive, abstract representation of physical space. The mapping between parameter space and physical space is achieved through the Jacobian matrix and its inverse. This is something widely utilized in FEM as well.

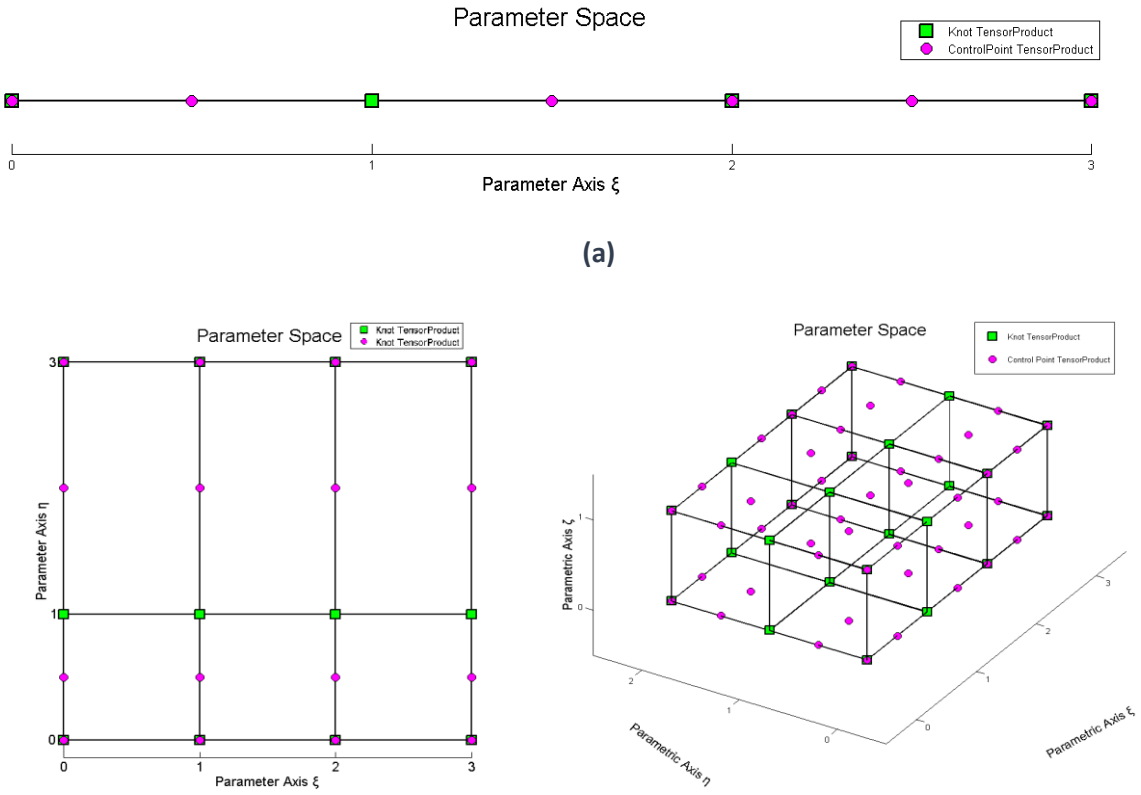


Figure 1.6.
Parameter space in 1D, 2D and 3D applications respectively.

The illustration of basis functions in the parameter space allows for a better understanding of concepts such as support, control point coordinates and the role of knots in basis function creation. In parameter space we usually also plot the basis and shape functions as in this space it is easy to see their support and observe their properties with clarity. Each knot marks the beginning and the end of a basis function domain. By “domain” we mean the area in which the basis function is non-zero, as all basis functions are defined throughout the parameter space, but are non-zero only in specific knot spans. Basis functions sharing the same domain are overlapping in parameter space and controlling a common part of the entity in the physical space.

1.3.3.3 Index Space

Index space is a representation of the model with respect to knot values. It is a line in 1D, containing the corresponding knot values in equally spaced positions. This space focuses upon the sequence of knot values rather than their actual numerical content.

Index space describes the contribution of each knot value to the creation of a certain B-Spline basis function. This helps identify the level of interconnection between basis functions and the knot value support of each function.

Control points are also evaluated in the index space. In fact, control points are defined as the center of the support of knot value spans.

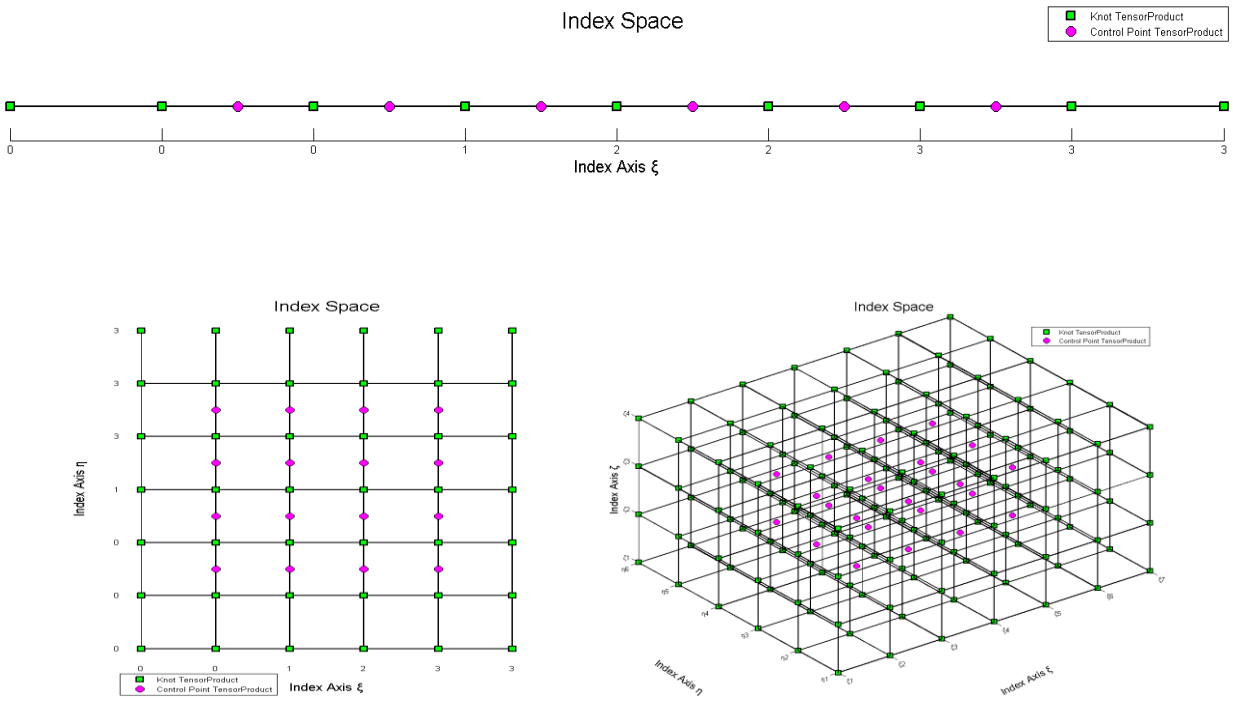


Figure 1.7.
Index space in 1D, 2D and 3D applications respectively.

Expansion to 2D or 3D leads to the creation of rectangles or cuboids respectively. Due to tensor product properties, everything mentioned about 1D extends and applies to both 2D and 3D. Thus, index space provides information that can contribute to the comprehension of a complex representation.

Index space is a line in 1D, a rectangle in 2D and a cuboid in 3D as it is shown in **Fig. 1.7**.

2. B-SPLINES AND NURBS

2.1 B-SPLINE BASIS FUNCTIONS

Given a sequence of non-decreasing numbers, $\Xi = \{\xi_1\ \xi_2\ \dots\ \xi_{n+p}\ \xi_{n+p+1}\}$, we can evaluate the B-Spline basis functions at $\xi \in \left[\xi_1, \xi_{n+p+1}\right]$ using the Cox-de Boor recursive formula (2.3):

First, for degree $p = 0$ (piecewise constant, box B-Spline)

$$N_{i,0}(\xi) = \begin{cases} 1, & \text{if } \xi_i \le \xi < \xi_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

The piecewise constant does not include the right edge ξ_{i+1} in order to ensure partition of unity, as the next basis function begins at that edge. The last function, however, includes both left and right edge, in order to be defined for the whole knot span.

$$N_{n+p,0}(\xi) = \begin{cases} 1, & \text{if } \xi_{n+p} \le \xi \le \xi_{n+p+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

Afterwards, for degree $p = 1, 2, \dots$ (with the assumption of $\frac{0}{0} \doteq 0$):

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} \cdot N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} \cdot N_{i+1,p-1}(\xi) \tag{2.3}$$

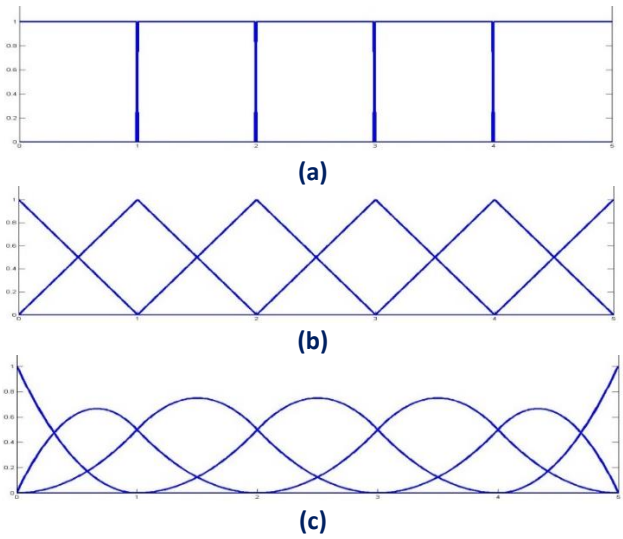


Figure 2.1.
Representation of B-Splines of degree **(a)** p= 0, **(b)** p=1and **(c)** p= 2
(Iakovos Antonios 2015).

2.2 KNOT VALUE VECTOR

Knots define the boundaries of the model's basis functions. They represent "switches" which turn "on" or "off" a certain piece of a B-Spline basis function. In order to acquire the knots and the basis functions, a knot vector must be defined.

A knot vector Ξ is usually defined in bibliography as a set of ξ_i coordinates, with:

$$\xi_i \leq \xi_{i+1}$$

It can contain the same number multiple times and generates the basis in a unique way.

In order to improve efficiency and communication between the readers of this thesis, we define as

- "Knot value vector": the whole set of non-decreasing coordinates ("knot values").
- "Knot vector": the set of unique coordinates ("knots").

For example, a knot value vector could be

{0 1 1 1 2 2 3 3}

where $0, 1, 1, 1, 2, 2, 3, 3$ are the separate knot values.

The corresponding knot vector is:

{0 1 2 3}

where 0,1,2,3 are the separate knots.

Let p be the polynomial degree of the basis function. If the first and the last knots are repeated p+1 times, the knot value vector is considered "open", because it has C^{-1} continuity on the edges, creating an open curve that is interpolatory at these points. If knot values are equally spaced, the knot value vector is considered "uniform". In CAD community, non-uniform, open knot value vectors are widely used. An example of a uniform knot value vector with polynomial degree p= 2 is:

{0 0 0 0.5 1 1.5 2 2.5 3 3 3}

A knot value vector may contain integers or decimals. In fact, the actual numerical content of knot values is of no importance. What matters is the relative distance between them. This means a knot value vector can be multiplied by any number, or have a number added to every knot value and the resulting basis would still be the same.

2.3 CONTROL POINTS COEFFICIENTS

Control points exist in all three spaces. Their parametric coordinates are defined as the center of the support in the index space. Recall that for the i^{th} basis function of order p the support is $[\xi_i,\xi_{i+p+1})$. The support contains p+1 knot value spans, therefore p+2 knot values (including the right boundary value ξ_{i+p+1}). For even degrees, the center of the support in the index space lies between two sequential knot values, i+p/2 and i+p/2+1. As a result, the coordinate of the control point is defined as the average of these knot values:

$$\xi_{CP} = 0.5 \cdot \left(\xi_{i+\frac{p}{2}} + \xi_{i+\frac{p}{2}+1} \right) \tag{2.4}$$

which means that a control point of even degree can either be on a knot, or in the middle of a knot span.

For odd degrees, the center of the support is the knot value i+(p+1)/2. Therefore, for odd degrees, control points are always coincident with knots.

2.4 B-SPLINE FULL TENSOR PRODUCT PROPERTY

B-Spline basis functions are of full-tensor product nature. Consequently, it is easy to combine B-Splines across different directions, in order to evaluate a multi-directional B-Spline shape function.

2D B-Spline shape functions can be evaluated as tensor product of basis functions $\mathrm{N}_{i,p}(\xi)$ and $\mathrm{M}_{j,q}(\eta)$:

$$\mathrm{R}_{i,j}^{p,q}(\xi,\eta) = \mathrm{N}_{i,p}(\xi) \cdot \mathrm{M}_{j,q}(\eta) \tag{2.5}$$

3D B-Spline shape functions are a tensor product of basis functions on three directions, $\mathrm{N}_{i,p}(\xi)$, $\mathrm{M}_{j,q}(\eta)$ and $\mathrm{L}_{k,r}(\zeta)$:

$$\mathrm{R}_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = \mathrm{N}_{i,p}(\xi) \cdot \mathrm{M}_{j,q}(\eta) \cdot \mathrm{L}_{k,r}(\zeta) \tag{2.6}$$

It is easily proved that all properties of B-Spline basis functions in one direction are inherited by the multi-directional shape functions. As a result, comprehension of univariate B-Spline basis function properties and techniques is very important for understanding multivariate complex geometries.

2.5 B-SPLINE BASIS FUNCTION PROPERTIES

B-Spline basis functions possess the following important properties:

- Local support:

$$N_{i,p}(\xi) = 0 \ \forall \xi \notin \left[\xi_i, \xi_{i+p+1}\right)$$

- In any given knot span, at most p+1 functions of order p are non-zero.
- Non-negativity:

$$N_{i,p}(\xi) \geq 0 \ \ \forall \ \xi, i, p$$

- Partition of unity:

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1 \ \forall \ \xi, p$$

- C^{p-m} Continuity across knots with m multiplicity.
- $N_{i,p}(\xi)$ has exactly one maximum value, except for p= 0.
- A non-periodic knot value vector that produces n functions of order p has n+p+1 knot values.
- Every B-Spline basis function shares support with 2p B-Splines.

2.5.1 Local support

Local support means that basis functions are non-zero only in certain knot spans in parameter space. This can be expressed by

$$N_{i,p}(\xi) = 0, \forall \xi \notin \left[\xi_i, \xi_{i+p+1}\right) \tag{2.7}$$

Local support is a result of the recursive character of B-Splines. For the creation of a B-Spline function of degree p, two consecutive B-Spline functions of order p-1 are used. For the creation of those consecutive basis functions, three consecutive functions of order p-2 are needed. Inductively, p+1 consecutive box basis functions are required.

Each box function has a support of one knot value span. As a result, the support of the final basis function is defined by the union of the supports of the box functions, hence $p+1$ consecutive knot value spans.

In **Fig. 2.2**, the recursive character of B-Splines is presented. Linear functions are combined for the evaluation of quadratic (p= 2) basis functions. Bear in mind that some linear and box functions are zero across the entire domain, but still contribute to the evaluation of the next-order B-Splines.

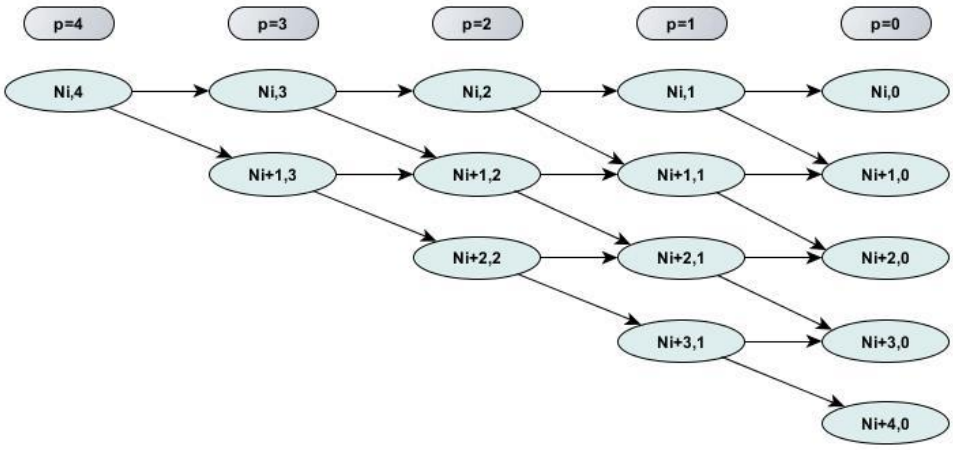


Figure 2.2.
Basis function generation (Cox-de Boor recursive formula).

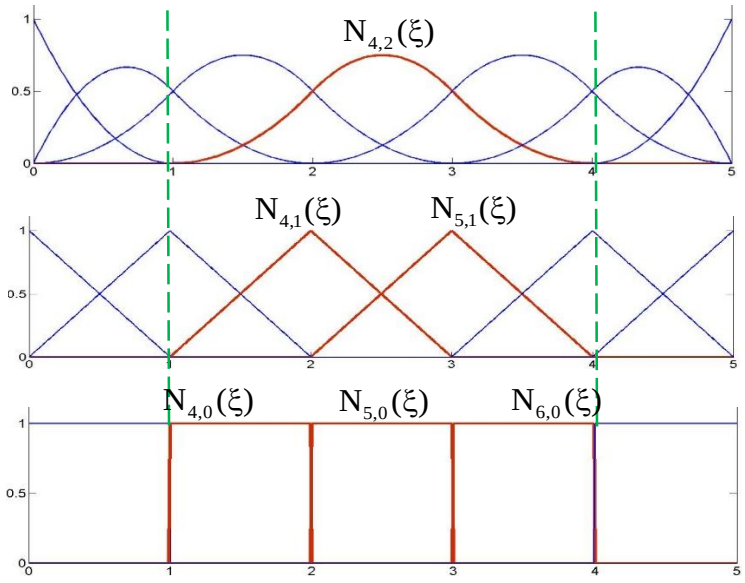


Figure 2.3.
B-Splines of degree 0, 1, 2 from bottom to top respectively
(Iakovos Antonios 2015)

For example, $N_{4,2}(\xi)$ in **Fig. 2.3** is created from $N_{4,1}(\xi)$ and $N_{5,1}(\xi)$. They, in turn, are evaluated from $N_{4,0}(\xi)$, $N_{5,0}(\xi)$ and $N_{6,0}(\xi)$ respectively. The corresponding box functions are non-zero in the knot spans [1,2), [2,3) and [2,3), [3,4), thus creating the support [1,4) of $N_{4,2}(\xi)$.

In a similar fashion, a support can be defined with respect to knot values contributing to the creation of a B-Spline basis function. These are the $p+2$ knot values that are contained in the knot value span support of the function. Both knot value span support and knot value support are necessary for the understanding of isogeometric methods.

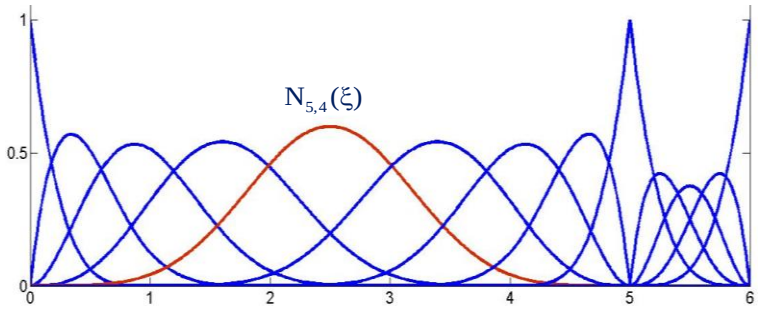


Figure 2.4.
B-Spline basis functions for knot value vector .Ξ={0 0 0 0 0 1 2 3 4 5 5 5 5 6 6 6 6 6}
1D Shape Function $N_{5,4}(\xi)$
(Iakovos Antonios 2015)

In **Fig. 2.4**, the degree is $p=4$, so each basis function has a support of p+1= 4+1= 5 knot value spans. The selected function $N_{5,4}(\xi)$, drawn in red, is non-zero only in the knot value spans [0, 1), [1,2), [2,3), [3,4) and [4,5). The p+2 = 6 knot values that define this function are {0,1,2,3,4,5}. There are no trivial spans in this B-Spline function, so it is considered a fully developed basis function.

B-Spline tensor product properties enable the immediate expansion of 1D properties to 2D and 3D B-Spline shape functions.

2.5.2 In any given knot span $[\xi_i,\xi_{i+1})$ at most (p+1) of the functions $N_{i,p}$ are nonzero

A box function that is non-zero in one knot span contributes to the evaluation of two consecutive B-Spline basis functions of order p=1. These two functions lead to the creation of three consecutive basis functions of order p=2. Inductively, a box function contributes to the creation of p+1 B-Spline basis functions of order p.

It applies from Cox-de Boor recursive formula that only one box function is non-zero across a selected knot span. As a result, only the corresponding p+1 B-Spline basis functions of order p can be non-zero in that specific knot span.

Therefore, at a non-trivial knot value span $[\xi_i, \xi_{i+1})$ only the basis functions $N_{i-p,p}(\xi)$, $N_{i-p+1,p}(\xi)$, …, $N_{i,p}(\xi)$ are non-zero. This is used efficiently in stiffness matrix formulation, in order to reduce computational cost.

In **Fig. 2.5**, two separate knot spans are examined.

For every knot span $p+1=3$ basis functions are non-zero.

For the first knot span, [0,1)=$[\xi_3,\xi_4)$ basis functions $N_{1,p}(\xi)$, $N_{2,p}(\xi)$, $N_{3,p}(\xi)$ are non-zero.

For the second knot span, [4,5)=[ξ_7,ξ_8) basis functions $N_{5,p}(\xi)$, $N_{6,p}(\xi)$, $N_{7,p}(\xi)$ are non-zero.

As a result, in 2D, at a given knot rectangle only the tensor products of the respective non-zero basis functions are non-zero, namely (p+1)(q+1) shape functions.

Inductively, in 3D, at a given knot cuboid the tensor products of the corresponding non-zero basis functions create (p+1)(q+1)(r+1) non-zero shape functions.

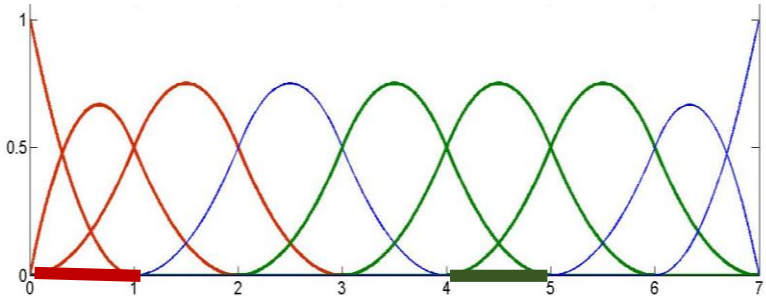


Figure 2.5.
B-Spline basis functions for knot value vector Ξ={0 0 0 1 2 3 4 5 6 7 7 7}. Basis functions that are non-zero in knot span [0,1) are drawn in red. Basis functions that are non-zero in knotspan [4,5) are drawn in green. (Iakovos Antonios and Karras Dimitrios 2015)

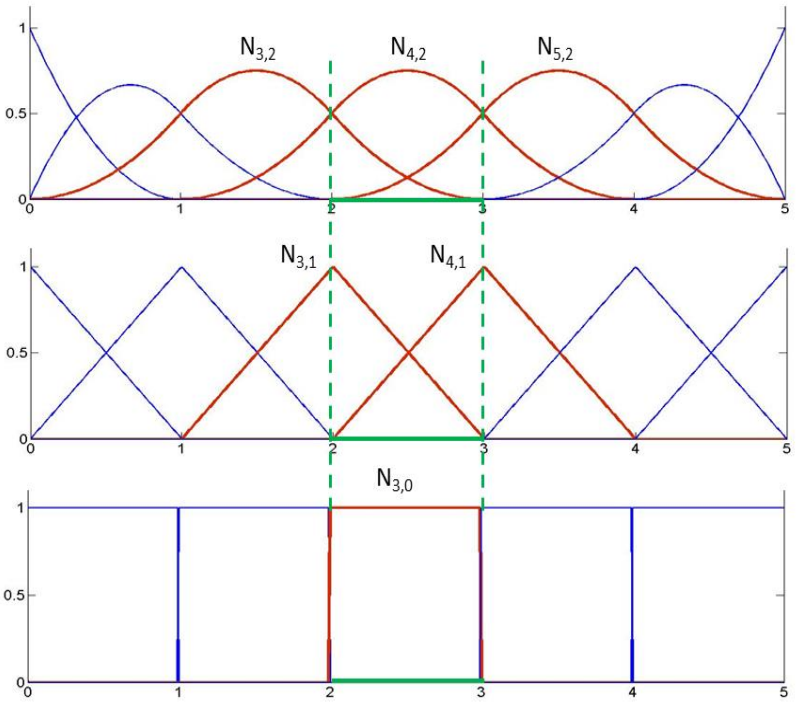


Figure 2.6.
Contribution of one box function to non-zero higher-order B-Spline basis functions across knot span [1,2) (Iakovos Antonios and Karras Dimitrios, 2015).

2.5.3 Non-negativity

It has been established that:

$$N_{i,p}(\xi) \geq 0 \quad \forall \ \xi, i, p \tag{2.8}$$

This is proven by induction on p. It is clearly true for p=0 and we assume it is true for p-1.

By definition

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

By the property of local support, either $\xi \notin \left[\xi_i, \xi_{i+p}\right)$ and $N_{i,p-1}(\xi)=0$ or $\xi \in \left[\xi_i, \xi_{i+p}\right)$ in which case $\xi-\xi_i \geq 0$ and with the assumption made that $N_{i,p-1}(\xi) \geq 0$, the first term $\frac{\xi-\xi_i}{\xi_{i+p}-\xi_i} N_{i,p-1}(\xi) \geq 0$. The same is true for the second term, $\frac{\xi_{i+p+1}-\xi}{\xi_{i+p+1}-\xi_{i+1}} N_{i+1,p-1}(\xi) \geq 0$. Thus $N_{i,p}(\xi) \geq 0$ and is valid for any i, p, ξ.

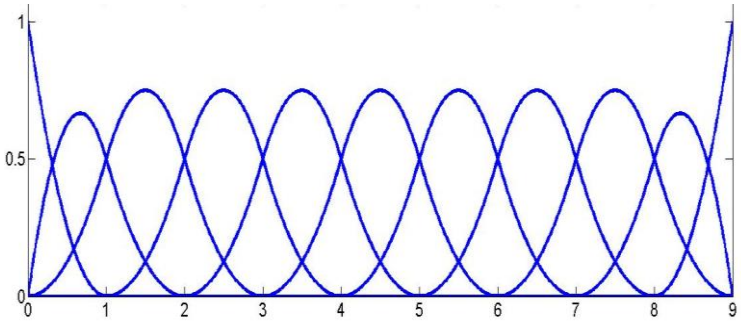


Figure 2.7.
B-Spline basis functions for knot value vector
Ξ={0 0 0 0 0 1 2 3 4 5 6 7 8 9 9 9 9 9}
(Iakovos Antonios 2015)

The degree in **Fig. 2.7** is p=2. As we see in the picture above, all the basis functions are positive for every ξ_i. This property is very important for isogeometric analysis and it does not apply in the classical shape functions of finite element analysis. Naturally, it is easier for the human mind to work with only positive values, so this attribute simplifies and encourages the understanding of B-Splines. Non-negativity applies for 2D and 3D shape functions as well.

2.5.4 Partition of unity

Partition of unity is expressed by

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \forall \xi, p \tag{2.9}$$

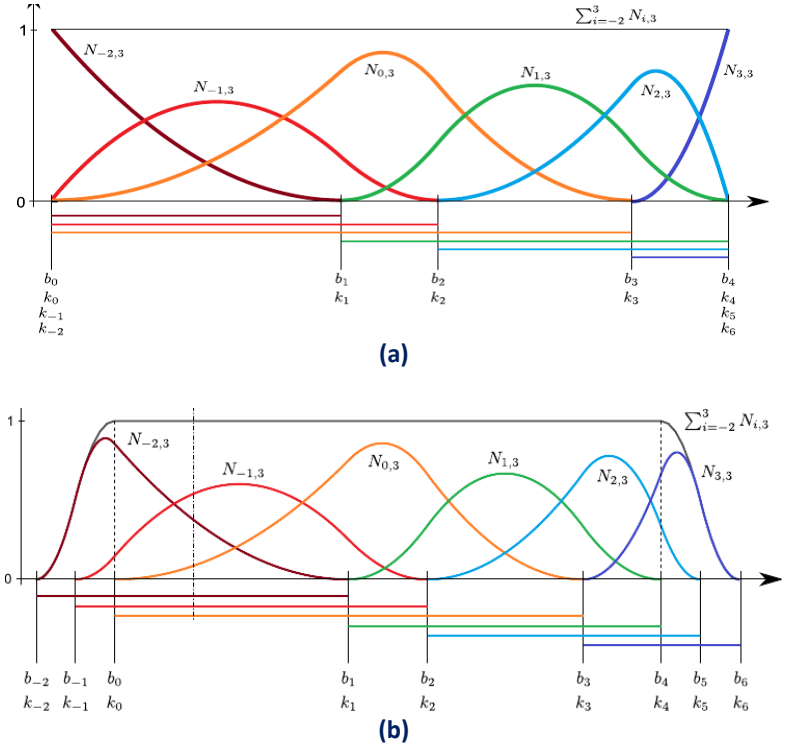


Figure 2.8.
Partition of unity in B-Spline basis for: **(a)** Open vector basis, **(b)** Non-open vector basis

Partition of unity is essential in any set of basis functions in any finite element scheme. The reason is simple. In isogeometric analysis as in the general case of finite elements, the displacement at a point C(ξ,η,ζ)={x,y,z} in the solid model, will be approximated by $\Sigma N_{i,p}(\xi)u_i$ where u_i are the displacements on the nodes or in the case of isogeometric analysis the displacements on the control points. If $\mathbf{u}_i = \bar{\mathbf{u}}$ for all the control points, we have rigid body displacements (property affine invariance of B-Spline curves) and if those in conjunction with the laws of the phenomenon we study cause no strain, it is clear that

$$\bar{u}\sum_{i=1}^{n} N_{i,p}(\xi) = \text{constant}.$$

In our case of isoparametric elements, where the same functions are used to describe geometry, it would be necessary that all points are translated by $\bar{u}$, thus

$$\bar{u}\sum_{i=1}^{n} N_{i,p}(\xi) = \bar{u} \Rightarrow \sum_{i=1}^{n} N_{i,p}(\xi) = 1.$$

Proof. We will prove it in an arbitrary knot span.

$\forall \xi \in \left[\xi_i, \xi_{i+1}\right)$,

$$\begin{aligned} \sum_{j=1}^{n} N_{j,p}(\xi) &= \sum_{j=i-p}^{i} N_{j,p}(\xi) = \sum_{j=i-p}^{i} \left(\frac{\xi-\xi_j}{\xi_{j+p}-\xi_j} N_{j,p-1}(\xi) + \frac{\xi_{j+p+1}-\xi}{\xi_{j+p+1}-\xi_{j+1}} N_{j+1,p-1}(\xi) \right) \\ &= \sum_{j=i-p}^{i} \frac{\xi-\xi_j}{\xi_{j+p}-\xi_j} N_{j,p-1}(\xi) + \sum_{j=i-p}^{i} \frac{\xi_{j+p+1}-\xi}{\xi_{j+p+1}-\xi_{j+1}} N_{j+1,p-1}(\xi) \end{aligned} \tag{2.10}$$

Changing variable in the second sum from (i-p) to (i-p+1) and considering that $N_{i-p,p-1} = N_{i+1,p-1} = 0$ for $\xi \in [\xi_i, \xi_{i+1})$:

$$\sum_{j=1}^{n} N_{j,p}(\xi) = \sum_{j=i-p+1}^{i} \left(\frac{\xi-\xi_j}{\xi_{j+p}-\xi_j} + \frac{\xi_{j+p}-\xi}{\xi_{j+p}-\xi_j} \right) N_{j,p-1}(\xi) = \sum_{j=i-p+1}^{i} N_{j,p-1}(\xi) \tag{2.11}$$

Applying the same procedure recursively:

$$\sum_{j=1}^{n} N_{j,p}(\xi) = \sum_{j=i-p+1}^{i} N_{j,p-1}(\xi) = \sum_{j=i-p+2}^{i} N_{j,p-2}(\xi) = \ldots = \sum_{j=i}^{i} N_{j,0}(\xi) = 1 \tag{2.12}$$

Partition of unity applies for multi-dimensional shape functions as well.

In 2D, partition of unity is expressed as

$$\sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q} = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) \cdot M_{j,q}(\eta) = 1 \tag{2.13}$$

which is a result of tensor product nature:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q} = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) \cdot M_{j,q}(\eta) = \left(\sum_{i=1}^{n} N_{i,p}(\xi) \right) \cdot \left(\sum_{j=1}^{m} M_{j,q}(\eta) \right) = 1 \tag{2.14}$$

In full analogy, 3D shape functions also possess partition of unity:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}^{p,q,r} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi)\cdot M_{j,q}(\eta)\cdot L_{k,r}(\zeta) = 1 \qquad (2.15)$$

Partition of unity is also a property of great importance for the shape functions that are used in finite element analysis.

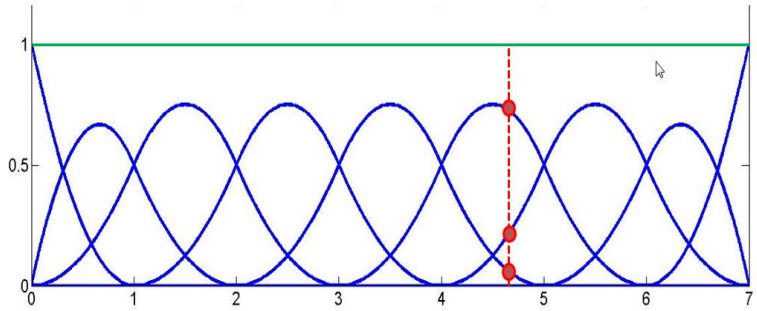


Figure 2.9.
B-Spline basis functions for knot value vector Ξ={0 0 0 0 1 2 3 4 5 6 7 7 7 7}.Sum of B-Spline function value at every ξ. Partition of unity.
(Iakovos Antonios 2015)

The degree for the B-Spline basis functions in **Figure 2.9** is p=3. The green horizontal line represents the sum of B-Spline values at the corresponding ξ, which, of course, is equal to 1 for every ξ.

For example, B-Spline values have been evaluated for ξ=4.81, with the following results for the four non-zero basis functions:

$$N_{5,3}(4.81) = 0.0011$$
$$N_{6,3}(4.81) = 0.2763$$
$$N_{7,3}(4.81) = 0.6340$$
$$N_{8,3}(4.81) = 0.0886$$

The above yields

$$\sum_{i=1}^{10} N_{i,3}(4.81) = 1$$

as expected.

2.5.5 C^{p-m} Continuity

At a knot with multiplicity m, $N_{i,p}(\xi)$ produces p-m continuous derivatives or, in other words, it has C^{p-m} continuity. Continuity less than C^0 is not acceptable for internal knots, which means they can be repeated up to p times. Bear in mind that as continuity decreases, B-Spline basis functions tend to be more "steep".

The polynomial order in **Fig. 2.10** is p=3. Continuity for a basis function is affected only by the corresponding knot values. Knot ξ=4 has multiplicity m=3 with respect to the knot vector. Its multiplicity with respect to the basis functions depends on each function's knot value support.

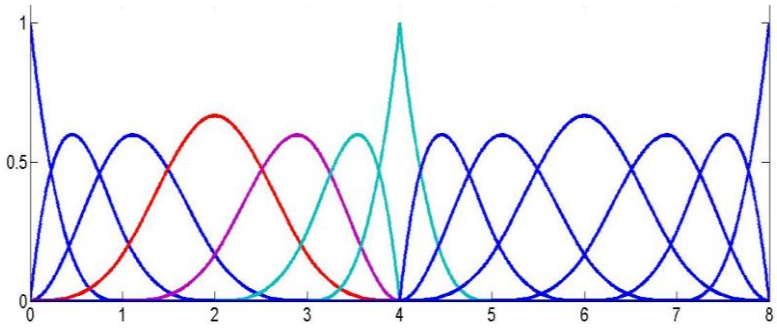


Figure 2.10.
B-Spline Basis functions for knot value vector
$\Xi = \{0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 4 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 8 \quad 8 \quad 8\}$
(Iakovos Antonios 2015)

$N_{4,3}(\xi)$, the red fully developed basis function has a knot value support of {0,1,2,3,4}. The knot ξ=4 has multiplicity m=1 with respect to the basis function $N_{4,3}(\xi)$,so this function is $C^{p-m} = C^{3-1} = C^2$ continuous in the entire domain.

The magenta basis function, $N_{5,3}(\xi)$ has a knot value support of {1,2,3,4,4}. The knot in question has multiplicity m=2 with respect to the basis function, which makes $N_{5,3}(\xi)$ $C^{p-m} = C^{3-2} = C^1$ continuous across ξ=4.

Knot value support for $N_{6,3}(\xi)$ and $N_{7,3}(\xi)$ which are depicted with cyan is {2,3,4,4,4} and {3,4,4,4,5} respectively. The multiplicity of the knot is m=3 for both basis functions. Therefore, these functions have $C^{p-m} = C^{3-3} = C^0$ continuity across ξ=4.

Observe that when a B-Spline basis function $N_{j,p}(\xi)$ has a knot $\xi_m$with multiplicity p in the center of the knot value support, it applies that $N_{j,p}(\xi_m)=1$.

In 2D and 3D cases, continuity per direction is obtained straight from the continuity of the corresponding one-directional B-Spline basis functions.

Observe that all functions of reduced continuity tend to be more "steep". This happens because they contain multiple knot values of the same knot, and therefore develop across

trivial spans. This “steepness” is the first indication that a basis function has reduced continuity.

Moreover, internal C^0 continuity produced exactly the same B-Spline basis functions as C^{-1} continuity on the edge of the knot vector. This means that at every knot with multiplicity m=p, only one function remains non-zero. Due to partition of unity, that function’s value at that knot is equal to 1.

In multi-dimensional functions, C^0 continuity across a knot requires the basis functions per all directions to be C^0 continuous at that point. In this case, the value of the multi-dimensional shape function across this knot is equal to 1.

2.5.6 Linear Independence

Given a finite number of distinct vectors $\{u_1\ u_2\ \dots\ u_n\}$ and scalars $\{a_1\ a_2\ \dots\ a_n\}$, the subset S of a vector space V is called linearly independent if the equation $a_1u_1+a_2u_2+\dots+a_nu_n=0$ leads to the unique solution $a_1=a_2=\dots=a_n=0$. Thus the subset is linearly independent if a linear combination of the vectors is the zero vector, only for $a_1=a_2=\dots=a_n=0$. The linear independence in isogeometric analysis applies for the basis functions $\{N_{1,p}(\xi)\ N_{2,p}(\xi)\ \dots\ N_{n,p}(\xi)\}$. The equation

$$\mathbf{a}_1 \cdot \mathbf{N}_{1,\mathrm{p}}(\xi) + \mathbf{a}_2 \cdot \mathbf{N}_{2,\mathrm{p}}(\xi) + \dots + \mathbf{a}_\mathrm{n} \cdot \mathbf{N}_{\mathrm{n},\mathrm{p}}(\xi) = 0 \qquad (2.16)$$

leads to $a_1u_1+a_2u_2+\dots+a_nu_n=0$. We can reach the conclusion that no B-Spline basis function can be expressed as a linear combination of the other B-Spline basis functions.

These linearly independent vectors form a basis for the vector space V. Some interesting attributes of such vectors include:

- The basis of the vector space V can be formed by different sets of linearly independent vectors. Any set can be used, provided that the vectors are linearly independent and all the properties above apply for every vector. Thus, in isogeometric analysis, we can choose different sets of B-Spline basis functions in order to represent the vector space.

- The number of vectors of any basis chosen is equal to the dimension of V, often represented as dim(V). In isogeometric analysis dim(V) is equal to the number of control points and, consequently, the number of basis functions used.

- Let there be a mass of vectors $\{u_1\ u_2\ \dots\ u_n\}$ which form the basis of a vector space with dim(V) and the numbers$\{a_1\ a_2\ \dots\ a_n\}$, called the coordinates of u. In isogeometric analysis, the basis of the vector space is the set of B-Spline basis functions $\{N_{1,p}(\xi)\ N_{2,p}(\xi)\ \dots\ N_{n,p}(\xi)\}$ and the numbers are the coordinates $\{X_1\ X_2\ X_3\}$ of the control points of the curve. A random vector u of the vector space V can be represented by the equation:

$$u = a_1 u_1 + a_2 u_2 + \ldots + a_n u_n \tag{2.17}$$

which in isogeometric analysis applies as

$$C(\xi) = X_1 \cdot N_{1,p}(\xi) + X_2 \cdot N_{2,p}(\xi) + \ldots + X_n \cdot N_{n,p}(\xi) \tag{2.18}$$

Any vector in V can be described as a linear combination of the basis. The numbers $\{a_1\ a_2\ \ldots\ a_n\}$ are the coordinates of u and they are unique for this specific u and this specific basis.

Vector space V in $\mathbb{R}$ is a mass of vectors with the properties below:

1. Commutativity of addition:

$$u + v = v + u, \forall u, v \in V \tag{2.19}$$

In isogeometric analysis this property applies as

$$C_1(\xi) + C_2(\xi) = C_2(\xi) + C_1(\xi) \tag{2.20}$$

2. Associativity of addition:

$$u + (v + w) = (u + v) + w, \forall u, v, w \in V \tag{2.21}$$

,

Respectively in isogeometric analysis,

$$C_1(\xi) + (C_2(\xi) + C_3(\xi)) = (C_1(\xi) + C_2(\xi)) + C_3(\xi) \tag{2.22}$$

3. Identity element of addition: There is an element $0 \in V$, the zero vector, so that

$$u + 0 = u,\ \forall u \in V$$

In isogeometric analysis the equation applies as

$$C(\xi) + 0 = C(\xi) \tag{2.23}$$

4. Inverse elements of addition: There is an element $-u \in V$ such that

$$u + (-u) = 0,\ \forall u \in V$$

The $-u$ is called the additive inverse of u. The equivalent in isogeometric analysis is

$$C(\xi)+(-C(\xi))=0 \tag{2.24}$$

5. Identity element of scalar multiplication: For the real number 1 , it applies:

$$1\cdot u=u,\ \forall u\in V$$

Respectively, in isogeometric analysis

$$1\cdot C(\xi)=C(\xi) \tag{2.25}$$

6. Distributivity of scalar multiplication with respect to vector addition: $\forall a\in\mathbb{R}$ and $\forall u,v\in V$ applies the relation

$$a\cdot(u+v)=a\cdot u+a\cdot v \tag{2.26}$$

In isogeometric analysis this equation applies as

$$a\cdot(C_1(\xi)+C_2(\xi))=a\cdot C_1(\xi)+a\cdot C_2(\xi) \tag{2.27}$$

7. Distributivity of scalar multiplication with respect to field addition: $\forall a,b\in\mathbb{R}$ and $\forall u\in V$ applies the relation

$$(a+b)\cdot u=a\cdot u+b\cdot u \tag{2.28}$$

which is implemented in isogeometric analysis as

$$(a+b)\cdot C(\xi)=a\cdot C(\xi)+b\cdot C(\xi) \tag{2.29}$$

8. Compatibility of scalar multiplication with field multiplication: $\forall a,b\in\mathbb{R}$ and $\forall u\in V$ applies the equation

$$a\cdot(b\cdot u)=(a\cdot b)\cdot u \tag{2.30}$$

Its equivalent in isogeometric analysis is

$$a\cdot(b\cdot C(\xi))=(a\cdot b)\cdot C(\xi) \tag{2.31}$$

B-Spline basis functions are indeed the basis for a vector space with dim(V), where n is the number of control points. Control points are the coordinates that transform the basis functions at any point in the given physical space. Conclusively,

$$\{N_{1,p}(\xi)\quad N_{2,p}(\xi)\quad \ldots\quad N_{n,p}(\xi)\} \tag{2.32}$$

is the basis of the vector space, while

$$\{X_1 \quad X_2 \quad \dots \quad X_n\} \tag{2.33}$$

are the coordinates for a vector $C(\xi)$. The familiar linear combination applies:

$$C(\xi) = X_1 \cdot N_{1,p}(\xi) + X_2 \cdot N_{2,p}(\xi) + \dots + X_n \cdot N_{n,p}(\xi) \tag{2.34}$$

Due to linear independence and vector space properties, it is understood that for a specific set of basis functions, only one set of control points can yield the appropriate geometry. If one wants to change the basis functions, the control points have to be shifted accordingly.

2.6 B-Spline Basis Function Derivatives

Basis function derivatives are widely used in isogeometric analysis. Deformation and stiffness matrices are built upon the derivatives of shape functions. As a result, the distribution of stresses and strains across the model is based on those derivatives. The derivatives of B-Splines, as obtained from the recursive formula, are represented as a linear combination of previous polynomial order basis functions :

$$\frac{d}{d\xi} N_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} \cdot N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \cdot N_{i+1,p-1}(\xi) \tag{2.35}$$

This leads to a generalized equation for the k^{th} derivative:

$$\frac{d^k}{d\xi^k} N_{i,p}(\xi) = \frac{p!}{(p-k)!} \cdot \sum_{j=0}^{k} a_{k,j} \cdot N_{i+j,p-k}(\xi) \tag{2.36}$$

$$a_{0,0} = 1 \tag{2.37}$$

$$a_{k,0} = \frac{a_{k-1,0}}{\xi_{i+p-k+1} - \xi_i} \tag{2.38}$$

$$a_{k,j} = \frac{a_{k-1,j} - a_{k-1,j-1}}{\xi_{i+p+j-k+1} - \xi_{i+j}}, \quad j=1, \dots, k-1, \tag{2.39}$$

$$a_{k,k} = \frac{-a_{k-1,k-1}}{\xi_{i+p+1} - \xi_{i+k}} \tag{2.40}$$

The partial derivatives of two-directional B-Spline shape functions can be easily obtained by application of the quotient rule:

$$\frac{\partial}{\partial \xi} R_{i,j}^{p,q}(\xi,\eta) = \left(\frac{d}{d\xi} N_{i,p}(\xi) \right) \cdot M_{j,q}(\eta) \tag{2.41}$$

$$\frac{\partial}{\partial\eta}R_{i,j}^{p,q}(\xi,\eta)=N_{i,p}(\xi)\cdot\left(\frac{d}{d\eta}M_{j,q}(\eta)\right) \tag{2.42}$$

3D Shape function derivatives per direction can be obtained in the same manner

$$\frac{\partial}{\partial\xi}R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)=\left(\frac{d}{d\xi}N_{i,p}(\xi)\right)\cdot M_{j,q}(\eta)\cdot L_{k,r}(\zeta) \tag{2.43}$$

$$\frac{\partial}{\partial\eta}R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)=N_{i,p}(\xi)\cdot\left(\frac{d}{d\eta}M_{j,q}(\eta)\right)\cdot L_{k,r}(\zeta) \tag{2.44}$$

$$\frac{\partial}{\partial\zeta}R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)=N_{i,p}(\xi)\cdot M_{j,q}(\eta)\cdot\left(\frac{d}{d\zeta}L_{k,r}(\zeta)\right) \tag{2.45}$$

2.7 B-SPLINE GEOMETRIES

B-Spline geometries can be curves, surfaces or solids and are created with the combination of B-Spline shape functions and their corresponding control points.

2.7.1 B-Spline Curve.

A p^{th} degree curve is defined by

$$(x,y,z)=C(\xi)=\underset{(1xn)}{\left\{N_{i,p}(\xi)\right\}^{T}}\underset{(n\,x\,3)}{\left\{P_i\right\}}=\sum_{i=1}^{n}N_{i,p}(\xi)\underset{(1x3)}{\left\{P_i\right\}}\quad,\quad\xi_1\le\xi\le\xi_{n+p+1} \tag{2.46}$$

where the $\{P_i\}$ are the control points' Cartesian coordinates of the curve and $\{N_{i,p}(\xi)\}$ the p^{th} degree B-Spline basis functions defined on the non-periodic open knot value vector $\Xi=\{\xi_1, \xi_2, \ldots, \xi_{n+p}, \xi_{n+p+1}\}$.

2.7.2 B-Spline Surface.

A B-Spline surface is obtained by taking a bidirectional net of control points, two knot vectors Ξ and H with the respective polynomial degrees p, q and the products of the univariate B-Spline functions:

$$(x,y,z)=S(\xi,\eta)=\sum_{i=1}^{n}\sum_{j=1}^{m}R_{i,j}^{p,q}(\xi,\eta)\underset{(1x3)}{\left\{P_{i,j,k}\right\}}=\sum_{i=1}^{n}\sum_{j=1}^{m}N_{i,p}(\xi)N_{j,q}(\eta)\underset{(1x3)}{\left\{P_{i,j}\right\}} \tag{2.47}$$

2.7.3 B-Spline Solid.

In the same fashion, a B-Spline volume is obtained by taking a three directional net of control points, three knot vectors Ξ, H, Z with the respective polynomial degrees p, q, r and the products of the univariate B-Spline functions:

$$(x,y,z) = V(\xi,\eta,\zeta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)\left\{P_{i,j,k}\right\}_{(1x3)} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi)N_{j,q}(\eta)N_{k,r}(\xi)\left\{P_{i,j,k}\right\}_{(1x3)} \quad (2.48)$$

2.8 POINT INVERSION

Note that the inverse procedure, finding the parametric coordinates ξ, η, ζ of a point (x,y,z) in physical space, is more difficult and may require an iterative procedure.

2.9 B-SPLINE CURVE PROPERTIES

B-Spline curve entities have the following properties, as obtained from Piegl, Tiller:

- B-Spline curves are a generalization of Bezier curves.
- C(ξ) is a piecewise polynomial curve.
- Each basis function corresponds to a certain control point.
- The first and last control point, as well as internal control points corresponding to C^0 continuous basis functions, are interpolatory to the curve.
- B-Spline curves possess strong convex hull property.
- Moving a control point X_i only changes part of the curve.
- The control polygon represents a piecewise linear approximation to the curve.
- It is possible to use multiple control points with the same coordinates.
- Any transformation applied to the curve can be applied directly at the control points. This property is known as “affine invariance” or “affine covariance”.
- In every knot span, at most p+1 control points contribute to the definition of the curve, corresponding to the p+1 non-zero basis functions.
- Since C(ξ) is a linear combination of $N_{i,p}(\xi)$, curve continuity and differentiability are obtained straight from the basis functions.
- No line has more intersections with the curve, than with the control polygon.

2.9.1 Generalization of Bezier curves

B-Spline curves are a generalization of Bezier curves. Given an open knot vector and n=p+1 control points, a Bezier curve is produced. Bezier curves were utilized in CAD methods before B-Splines. **Fig. 2.11** presents a Bezier curve with p=3, an open knot value vector {0 0 0 0 1 1 1 1} and n=p+1=4 control points. Bezier curves are B-Spline curves defined in only one knot span. As a result, every basis function is non-zero across the entire parameter space and control points affect the shape of the entire curve. Bezier surfaces are a special case of one-rectangle B-Spline surfaces and Bezier solids a special case of one-cuboid B-Spline solids as well.

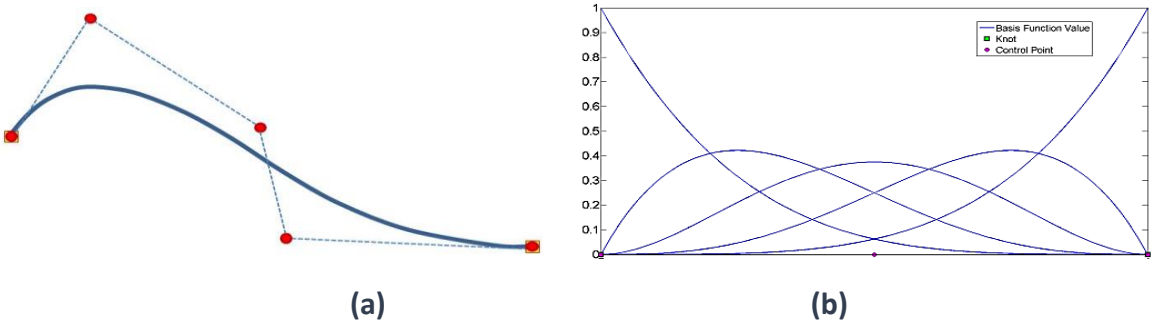


Figure 2.11.
Bezier curve in **(a)** physical space and **(b)** basis functions in parameter space. Knot value vector: Ξ = { 0 0 0 0 0 1 1 1 1 1 }
(Iakovos Antonios 2015)

2.9.2 C(ξ) is a piecewise polynomial curve

We can see that through the definition of the curve,

$$(x,y,z) = C(\xi) = \underset{(1xn)}{\left\{N_{i,p}(\xi)\right\}^T} \underset{(n\,x\,3)}{\left\{P_i\right\}} = \sum_{i=1}^{n} N_{i,p}(\xi) \underset{(1x3)}{\left\{P_i\right\}} \quad , \quad \xi_1 \le \xi \le \xi_{n+p+1} \qquad (2.49)$$

We know that the B-Spline basis functions $N_{i,p}(\xi)$ are piecewise polynomials which multiplied by coefficients $\{P_i\}$ and then summed, also result in a piecewise polynomial of ξ, the curve C(ξ). Of course the tensor product surfaces and solids from combining B-Spline basis functions along different directions ξ, η, ζ are also piecewise polynomials in respect to those directions. In general they are piecewise polynomials of ξ, η and ζ.

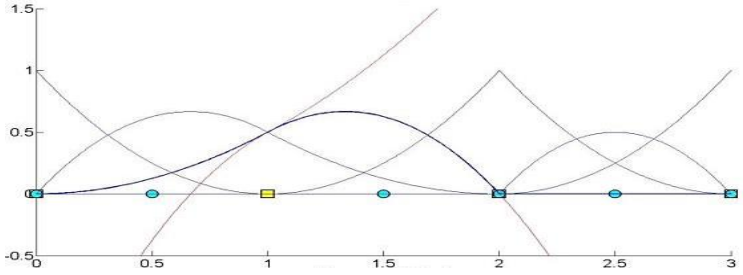
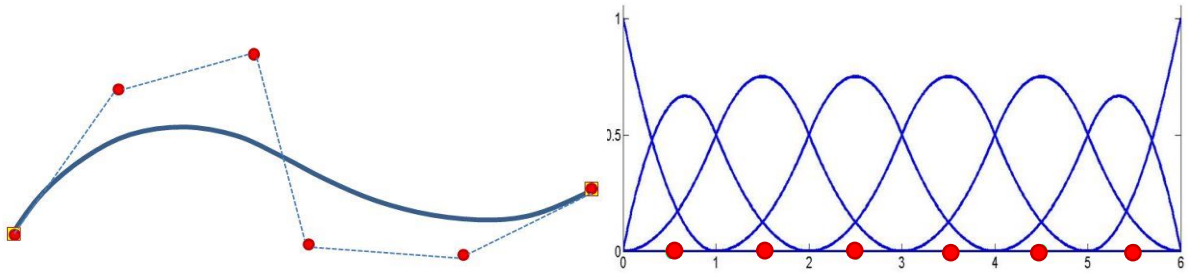


Figure 2.12.
Piecewise polynomials that form a B-Spline basis function.Knots represent the boundaries of the pieces.

2.10 Control Point – Basis Function Correspondence

Each basis function corresponds to a certain control point. There are n basis functions and n control points in a B-Spline curve.



(a) **(b)**

Figure 2.13.
(a) Physical Space and **(b)** basis functions with the corresponding control points.
(Iakovos Antonios 2015)

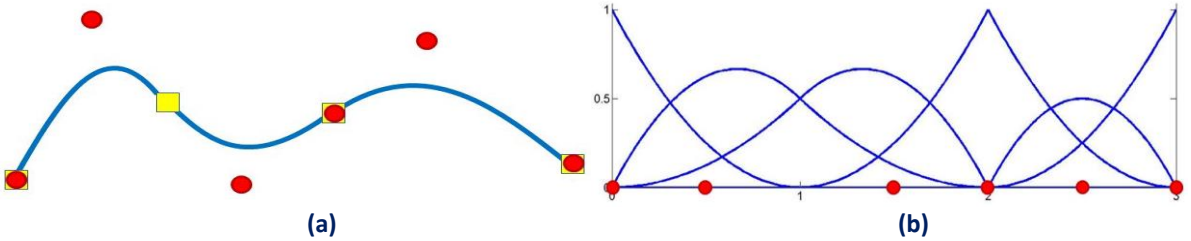
In **Fig. 2.13**, control points are represented both in parameter and physical space. Each point controls a specific basis function. This property also applies for multiple directions. Every control point of the surface or the solid is tensor product of a control point in directions ξ, η and ζ. By extension, the corresponding B-Spline is tensor product of the basis functions. In the most general case, there are (n x m x l) control points and basis functions.

2.10.1 Interpolation to the Curve

The endpoint interpolation property states that the first and last control point of the curve lay upon the curve, on its first and last point. We will support this argument and also augment it with the statement that for every knot of C^0 continuity internal to the knot vector a control point also lies upon the curve.

At points of C^0 continuity internally to the knot vector and C^{-1} continuity at the edges of the knot vector the shape functions are all zero except one that is 1.

$$C(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) \underset{(1x3)}{\{P_i\}} \tag{2.50}$$



(a) **(b)**

Figure 2.14.
Control point interpolation.**(a)**B-Spline curve and **(b)** the reciprocal basis functions

In **Fig. 2.14**, the first and the last control point, which have C^{-1} continuity, are interpolatory to the curve. This can be explained with the help of the equation of the curve:

$$C(\xi) = \sum_{i=1}^{n} \left\{ N_{i,2}(\xi) \cdot X_i \right\}$$

For $\xi = 0$, it applies that:

$$C(0) = \sum_{i=1}^{n} N_{i,2}(0) \cdot X_i$$

where,

$$N_{1,2}(0) = 1$$

$$N_{i,2}(0) = 0, \; i = 2,...,6$$

so,

$$C(0) = N_{1,2}(0) \cdot X_1 = X_1$$

And for $\xi = 3$:

$$C(3) = \sum_{i=1}^{n} N_{i,2}(3) \cdot X_i$$

$$N_{6,2}(3) = 1$$

$$N_{i,2}(3) = 0, \; i = 1,...,5$$

so,

$$C(3) = N_{6,2}(3) \cdot X_6 = X_6$$

Likewise, the internal control point, with C^0 continuity across $\xi = 2$ is interpolatory to the curve because:

$$C(2) = \sum_{i=1}^{n} \left\{ N_{i,2}(2) \cdot X_i \right\}$$

$N_{4,2}(2) = 1$, as this is the only non-zero basis function across $\xi = 2$

$$N_{i,2}(2) = 0, \; i \neq 4$$

so,

$$C(2) = N_{4,2}(2) \cdot X_4 = X_4$$

Observe that both the form of the curve and the form of the basis functions indicate that this geometry could be represented by two different sets of knot vectors and control points,

with absolutely no deflections from the current representation. This will be examined thoroughly later.

Interpolation also applies for surfaces and solids, when appropriately reduced continuity is used for all directions at a knot. C^{-1} continuity is required for external knots and C^0 for internal.

In **Fig. 2.15**, $N_{5,2}(\xi)$ at axis ξ and $M_{6,2}(\eta)$ at axis η have C^0 Continuity. Therefore the control point corresponding to $R^{2,2}_{5,6}(\xi,\eta)$ is interpolatory to the surface. The external control points with C^{-1} continuity at both directions are also interpolatory to the surface.

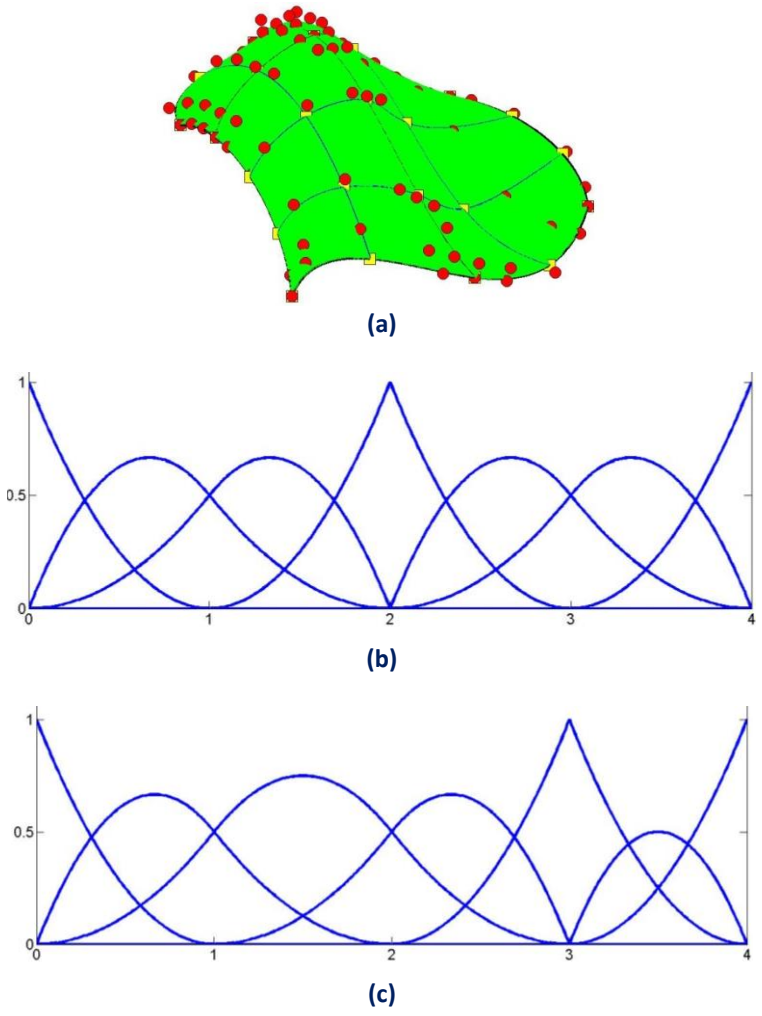


Figure 2.15.
(a) B-Spline surface and **(b)** Basis functions axis ξ, **(c)** Basis functions in axis η .
(Iakovos Antonios 2015)

In **Fig. 2.16** $N_{4,2}(\xi)$ at parametric axis ξ and $M_{4,2}(\eta)$ at parametric axis η have C^0 continuity, thus control points corresponding to the shape functions $R^{2,2,1}_{4,4,1}(\xi,\eta,\zeta)$ and

$R^{2,2,1}_{4,4,2}(\xi,\eta,\zeta)$, are interpolatory to the solid. As we can see, the external control points are also interpolatory to the solid, because continuity is reduced to C^{-1}.

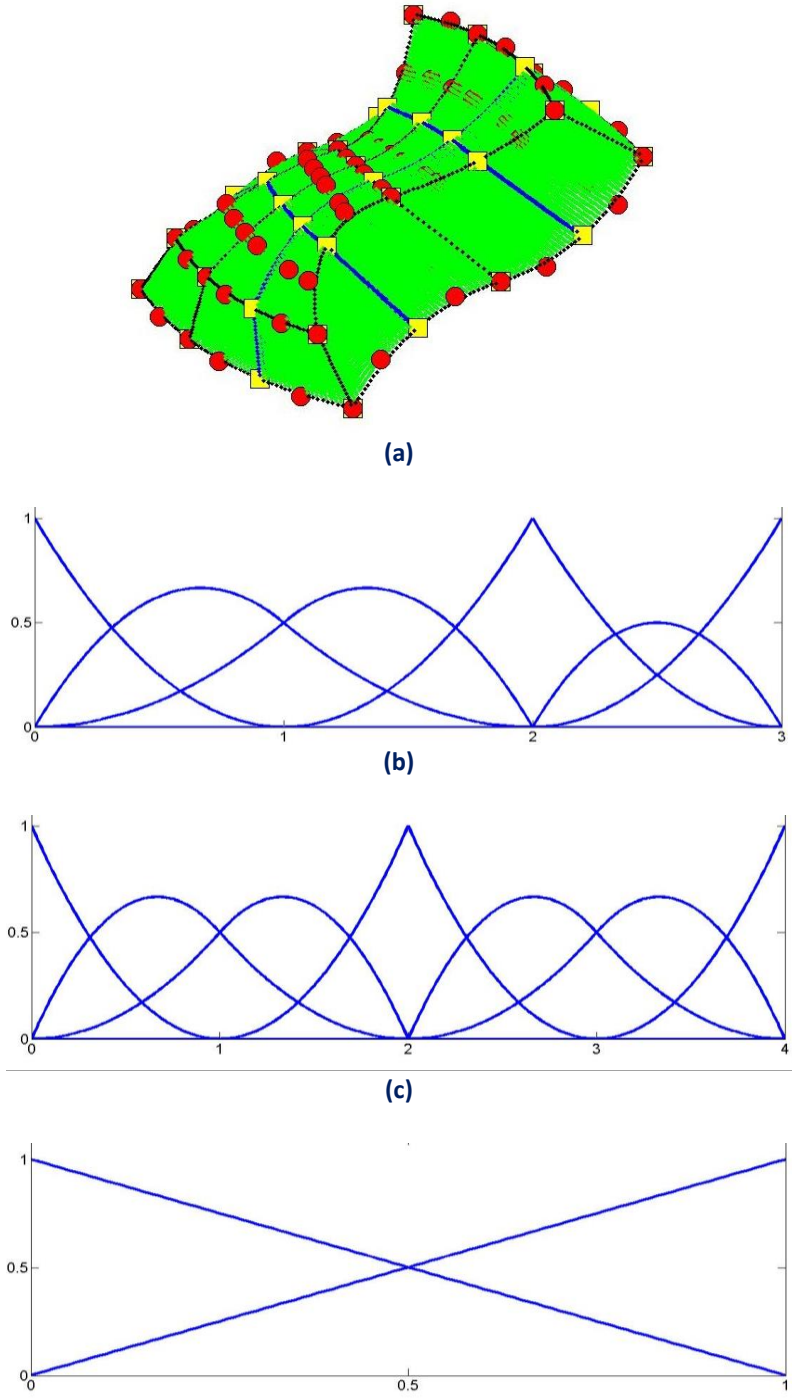


Figure 2.16.
(a) Solid in the physical space with **(b)** basis functions axis ξ, **(c)** basis functions axis η, **(d)** basis functions axis ζ. (Iakovos Antonios 2015)

2.10.2 B-Spline convex hull property

The strong convex hull property states that the curve is contained in the convex hull of its control polygon and more specifically that:

If $\xi \in \left[\xi_i, \xi_{i+1}\right)$ with $p \le i < n$, so that ξ is not in the starting and ending (p+1) trivial knot spans, then C(ξ) is in the convex hull of the control points $\mathbf{P}_{i-p}, \mathbf{P}_i$.

The strong convex hull property follows from the properties of non-negativity, partition of unity and local support of the B-Spline Basis functions.

In **Figure 2.17** we demonstrate the strong convex hull property at each knot span and over the whole curve.

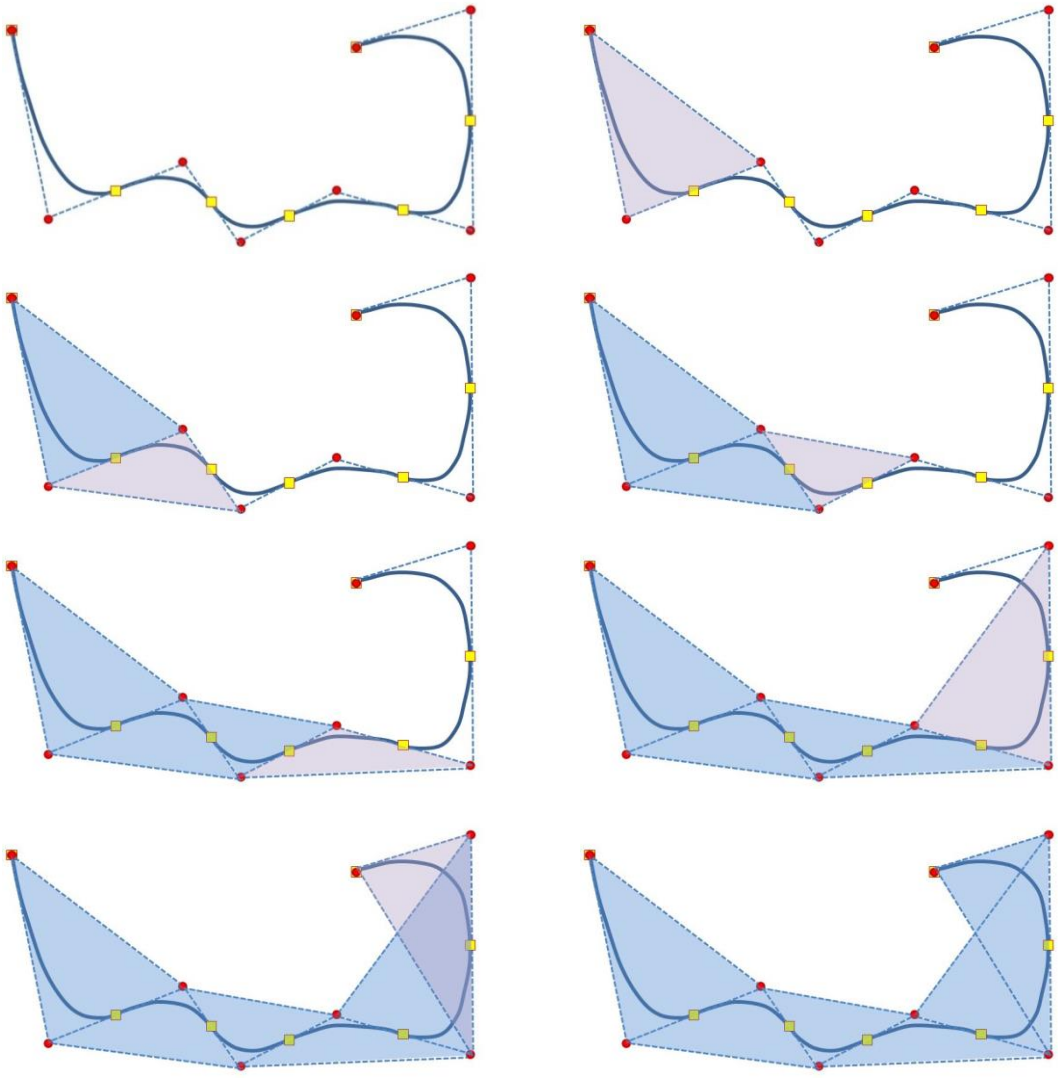


Figure 2.17.
Demonstration of the strong convex hull property in B-Spline curves. Knot Value Vector Ξ = { 0 0 0 1 2 3 4 4 4} (Iakovos Antonios 2015)

2.10.3 Control Point Local support

Moving a control point X_i only changes part of the curve, more specifically the part corresponding to the $[\xi_i, \xi_{i+p+1})$ knot value spans. This is a result of the local support of the corresponding B-Spline function.

Changing the i^{th} control point's Cartesian coordinate doesn't matter outside the interval $[\xi_i, \xi_{i+p+1})$ as only there the corresponding B-Spline $N_{i,p}(\xi)\neq 0$. Thus outside $[\xi_i, \xi_{i+p+1})$ the contribution $N_{i,p}(\xi)\{P_i\}$ to the curve is zero whether we change the coordinate or not. So indeed, moving a control point P_i changes only a part of the curve, the mapping of the interval $[\xi_i, \xi_{i+p+1})$ to the physical space.

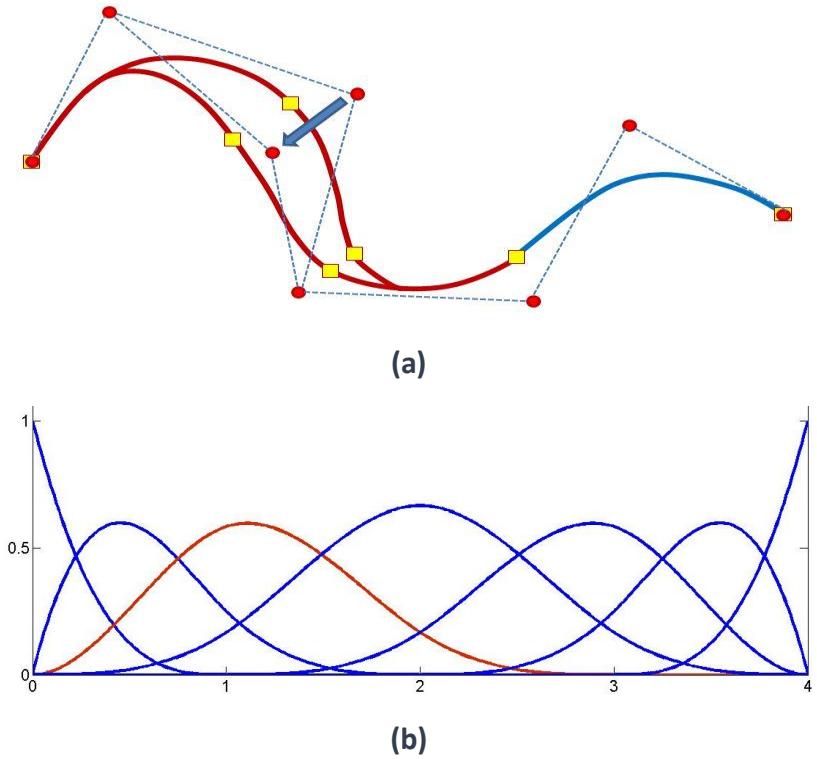


Figure 2.18.
The control point local support in **(a)** physical and **(b)** parameter space.Moving a control point affects only part of the curve.
(Iakovos Antonios 2015)

Every control point is associated with a basis function. Support of the control point is defined by the support of the corresponding basis function. Therefore, control points affect only part of the curve. In **Fig. 2.18**, a curve with p=2 and knot value vector {0 0 0 1 2 3 4 4 4}, is presented. The control point for i=3 is moved and this affects partially the entity.

In this example, $N_{3,2}(\xi)$ spans from ξ=0 to ξ=3. The knots act as boundaries of the support. For ξ=3, $N_{3,2}(\xi)$ is switched "off" and $N_{6,2}(\xi)$ is switched "on".

The 3rd B-Spline and the corresponding 3rd control point have support the interval [0,3) and that is the interval in the physical space that is influenced by the change of the control point's Cartesian coordinate. We notice that in the interval [3,4), out of the control points' support the curve remains intact.

Local support of control points is also expanded by tensor product properties. The local support of a multi-directional control point is the tensor product of the respective supports.

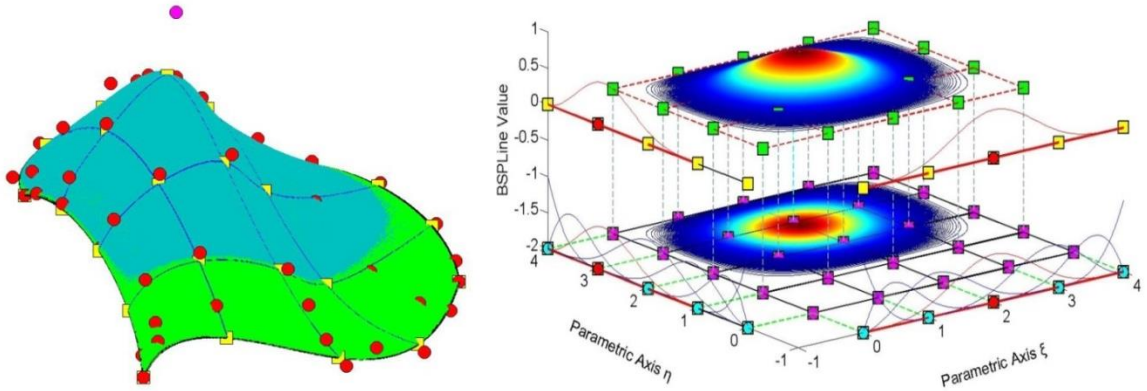


Figure 2.19.
Local support of 2D control point.Surface **(a)** in physical and **(b)** in parameter space.
(Iakovos Antonios 2015)

Fig. 2.19 represents the local support in the parametric axis ξ,η of $N_{4,2}(\xi)$ and $M_{3,2}(\eta)$ B-Spline curves respectively. In parametric axis ξ the local support expands throughout the axis, whereas in parametric axis η the basis function is switched "on" at the second knot span. The tensor product of the respective supports for ξ,η is represented in cyan. It spans across 4 x 3= 12 knot rectangles in total.

The same properties apply for B-Spline solids as well.

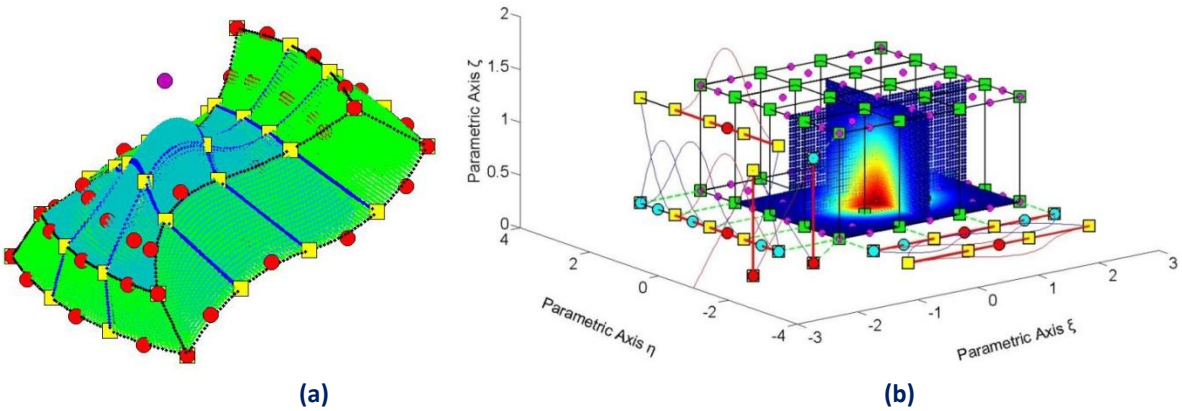


Figure 2.20.
Local support of a 3D control point.Solid **(a)** in physical and **(b)** in parameter space.
(Iakovos Antonios 2015)

Fig. 2.20 presents the local support of $N_{3,2}(\xi)$, $M_{4,2}(\eta)$ and $L_{1,1}(\zeta)$. In parametric axes ξ, ζ, the local support expands at all knot spans, whereas in the parametric axis η the local support spans between knots 1 and 4. The local support is displayed in cyan and it does not reach all the knot spans in parametric axis η. A total of 3 x 3 x 1 = 9 knot cuboids represent the support of the control point.

2.10.4 Control Polygon Approximation

The control polygon represents a piecewise linear approximation to the curve. Due to convex hull properties, refinement by knot insertion or order elevation brings the control polygon closer to the curve.

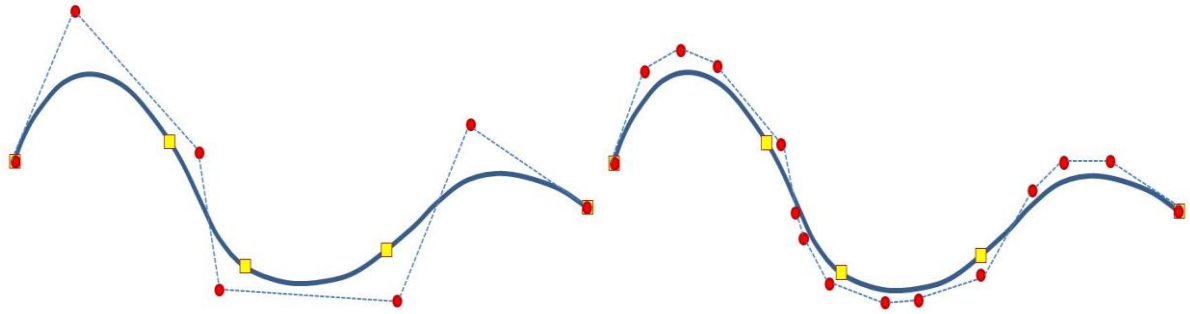


Figure 2.21
Control polygon approximation through refinement.
(Iakovos Antonios 2015)

In **Fig. 2.21** a curve of degree p=3 is designed. The control polygon already represents a linear approximation to the curve. When consecutive h- or p- refinements are applied, the control polygon is brought even closer to the curve. Refined control polygons provide a general idea of the form of the curve. This property also applies for multiple directions.

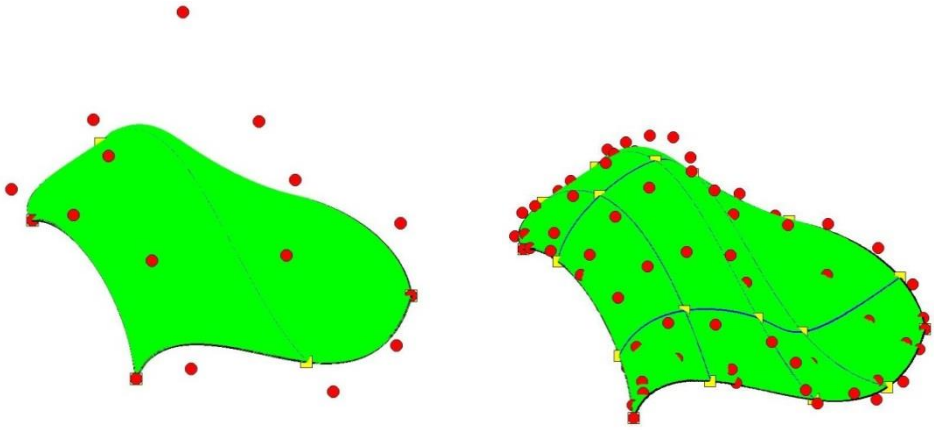


Figure 2.22.
Control Net approximation through Surface h-Refinement.
(Iakovos Antonios 2015)

For example, in **Fig. 2.22**, the refinement iterations made, brought the control net closer to the surface.

2.11 NON-UNIFORM RATIONAL B-SPLINES

2.11.1 Introduction

Non-uniform rational B-Spline (NURBS) is a mathematical model commonly used in computer graphics for generating and representing curves and surfaces. It offers great flexibility and precision for handling both analytic (surfaces defined by common mathematical formulae) and modeled shapes. NURBS are commonly used in computer-aided design (CAD), manufacturing (CAM), and engineering (CAE) and are part of numerous industry standards, such as IGES, STEP, ACIS, and PHIGS. NURBS tools are also found in various 3D modeling and animation software packages.

They can be handled efficiently by computer software and yet allow easy human interaction. NURBS surfaces are functions of two parameters, mapping a surface in three-dimensional space. The shape of the surface is determined by control points. NURBS surfaces can represent in compact form, simple geometrical shapes. T-splines and subdivision surfaces are more suitable for complex organic shapes ,as they reduce the number of control points twofold in comparison with the NURBS surfaces.

In general, editing NURBS curves and surfaces is highly intuitive and predictable. Control points are always either connected directly to the curve/surface, or act as if they were connected by a rubber band. Depending on the type of user interface, editing can be realized via an element's control points, which are most obvious and common for Bézier curves, or via higher level tools such as spline modeling or hierarchical editing.

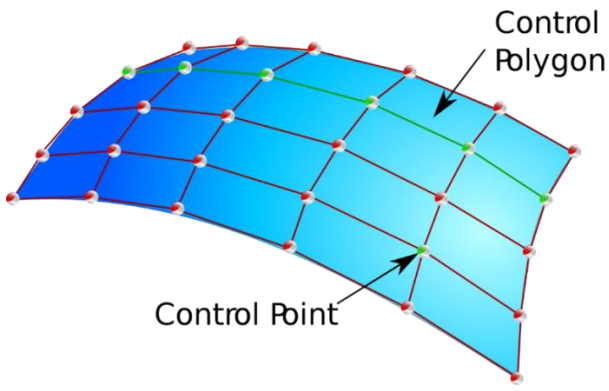


Figure 2.23.

Illustration of random NURBS surface along with its control polygon.Control points are depicted as small spheres

2.11.2 Basic Properties and advantages

A NURBS curve is defined by its order, a set of weighted control points and a knot vector. NURBS curves and surfaces are generalizations of both B-Splines and Bézier curves and surfaces. The primary difference is the weighting of control points, which makes NURBS curves rational (non-rational B-splines are a special case of rational B-Splines). By using a two-dimensional grid of control points, NURBS surfaces, planar patches and sections of spheres can be created. These are parametrized with two variables (typically called s and t or u and v). This can be extended to arbitrary dimensions to create NURBS mapping $R^n \rightarrow R^n$.

NURBS have numerous advantages in computational graphics and design that can be exploited in isogeometric analysis:

- They are invariant under affine transformations: operations like rotations and translations can be applied to NURBS curves and surfaces by applying them to their control points.

- They offer one common mathematical form for both standard analytical shapes (e.g., conics) and free-form shapes.

- They provide the flexibility to design a large variety of shapes.

- They reduce the memory consumption when storing shapes (compared to simpler methods).

- They can be evaluated reasonably quickly by numerically stable and accurate algorithms.

2.11.3 Concept

The projective B-Spline curve $C^w(\xi)$ is created from the projective 3D control points $X^w=\{X^w\ Y^w\ Z^w\}$.

Projection of the curve and control points on the plane z=1 produces the NURBS curve C(ξ) and the 2D control points:

$$X = \{X\ Y\}$$

where,

$$\{X_i \quad Y_i\} = \left\{ \frac{X_i^w}{Z_i^w} \quad \frac{Y_i^w}{Z_i^w} \right\}$$

The weights of the NURBS curve are defined as:

$$w = \{Z^w\}$$

In generalization, n-dimensional rational B-Splines are projections of (n+1) dimensional non-rational B-Splines.

2.11.4 NURBS Shape Functions

In order to evaluate NURBS shape functions, the weighting function is defined:

$$W(\xi) = \sum_{i=1}^{n} \{ N_{i,p}(\xi) \cdot w_i \}$$

In most engineering applications, weights have positive values. $W(\xi)$ is in fact the Z-coordinate of the projective B-Spline curve. Projective transformation is applied by dividing the other two coordinates of the B-Spline curve with the Z-coordinate. NURBS shape functions are calculated from

$$R_i^p(\xi) = \frac{N_{i,p}(\xi) \cdot w_i}{W(\xi)} = \frac{N_{i,p}(\xi) \cdot w_i}{\sum_{i'=1}^{n} \{ N_{i',p}(\xi) \cdot w_{i'} \}}$$

$R_i^p(\xi)$ are piecewise rational functions. The expression “the order of NURBS” refers to the order of the projective B-Spline curve.

NURBS shape functions in multiple directions can be obtained as tensor products of one-directional basis functions:

Shape functions for two directions:

$$R_{i,j}^{p,q}(\xi,\eta) = \frac{N_{i,p}(\xi) \cdot M_{j,q}(\eta) \cdot w_{ij}}{\sum_{i'=1}^{n} \sum_{j'=1}^{m} \{ N_{i',p}(\xi) \cdot M_{j',q}(\eta) \cdot w_{i'j'} \}}$$

$$W(\xi,\eta) = \sum_{i'=1}^{n} \sum_{j'=1}^{m} \{ N_{i',p}(\xi) \cdot M_{j',q}(\eta) \cdot w_{i'j'} \}$$

Similarly, we expand the equations in order to obtain tensor product 3D shape functions:

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)=\frac{N_{i,p}(\xi)\cdot M_{j,q}(\eta)\cdot L_{k,r}(\zeta)\cdot w_{ijk}}{\sum_{i'=1}^{n}\sum_{j'=1}^{m}\sum_{k'=1}^{l}\left\{N_{i',p}(\xi)\cdot M_{j',q}(\eta)\cdot L_{k',r}(\zeta)\cdot w_{i'j'k'}\right\}}$$

The weighting function is now defined as:

$$W(\xi,\eta,\zeta)=\sum_{i'=1}^{n}\sum_{j'=1}^{m}\sum_{k'=1}^{l}\left\{N_{i',p}(\xi)\cdot M_{j',q}(\eta)\cdot L_{k',r}(\zeta)\cdot w_{i'j'k'}\right\}$$

Observe that for $w_{ijk}=1,\ \forall\, i,j,k$, it applies that NURBS shape functions downgrade to B-Spline basis functions. Actually, NURBS entities are a generalization of B-Spline entities. All the B-Spline properties examined in this thesis apply for NURBS as well.

2.11.5 NURBS Shape Function Derivatives

Applying the quotient rule we get the expression for the derivatives of NURBS shape functions.

$$\frac{d}{d\xi}R_i^p(\xi)=\frac{\left(\frac{d}{d\xi}N_{i,p}(\xi)\right)W(\xi)-N_{i,p}(\xi)\left(\frac{d}{d\xi}W(\xi)\right)}{\left(W(\xi)\right)^2}w_i$$

In the case of more than one parametric direction we get the partial derivatives in the same way.

For 2D shape functions:

$$\frac{\partial}{\partial\xi}R_{i,j}^{p,q}(\xi,\eta)=\frac{\left(\frac{\partial}{\partial\xi}N_{i,p}(\xi)\right)W(\xi,\eta)-N_{i,p}(\xi)\left(\frac{\partial}{\partial\xi}W(\xi,\eta)\right)}{\left(W(\xi,\eta)\right)^2}w_{i,j}M_{j,q}(\eta)$$

$$\frac{\partial}{\partial\eta}R_{i,j}^{p,q}(\xi,\eta)=\frac{\left(\frac{\partial}{\partial\eta}M_{j,q}(\eta)\right)W(\xi,\eta)-M_{j,q}(\eta)\left(\frac{\partial}{\partial\eta}W(\xi,\eta)\right)}{\left(W(\xi,\eta)\right)^2}w_{i,j}N_{i,p}(\xi)$$

And for 3D shape functions

$$\frac{\partial}{\partial\xi}R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)=\frac{\left(\frac{\partial}{\partial\xi}N_{i,p}(\xi)\right)W(\xi,\eta,\zeta)-N_{i,p}(\xi)\left(\frac{\partial}{\partial\xi}W(\xi,\eta,\zeta)\right)}{\left(W(\xi,\eta,\zeta)\right)^2}w_{i,j,k}M_{j,q}(\eta)L_{k,r}(\zeta)$$

$$\frac{\partial}{\partial \eta} R^{p,q,r}_{i,j,k}(\xi,\eta,\zeta) = \frac{\left(\frac{\partial}{\partial \eta} M_{j,q}(\eta)\right) W(\xi,\eta,\zeta) - M_{j,q}(\eta)\left(\frac{\partial}{\partial \eta} W(\xi,\eta,\zeta)\right)}{\left(W(\xi,\eta,\zeta)\right)^2} w_{i,j,k} N_{i,p}(\xi) L_{k,r}(\zeta)$$

$$\frac{\partial}{\partial \zeta} R^{p,q,r}_{i,j,k}(\xi,\eta,\zeta) = \frac{\left(\frac{\partial}{\partial \zeta} L_{k,r}(\zeta)\right) W(\xi,\eta,\zeta) - L_{k,r}(\zeta)\left(\frac{\partial}{\partial \zeta} W(\xi,\eta,\zeta)\right)}{\left(W(\xi,\eta,\zeta)\right)^2} w_{i,j,k} N_{i,p}(\xi) M_{j,q}(\eta)$$

2.11.6 NURBS Entities

NURBS entities are created as a linear combination of NURBS shape functions, exactly the same way as B-Spline entities. The following is the equation for the creation of NURBS

2.11.6.1 Curves:

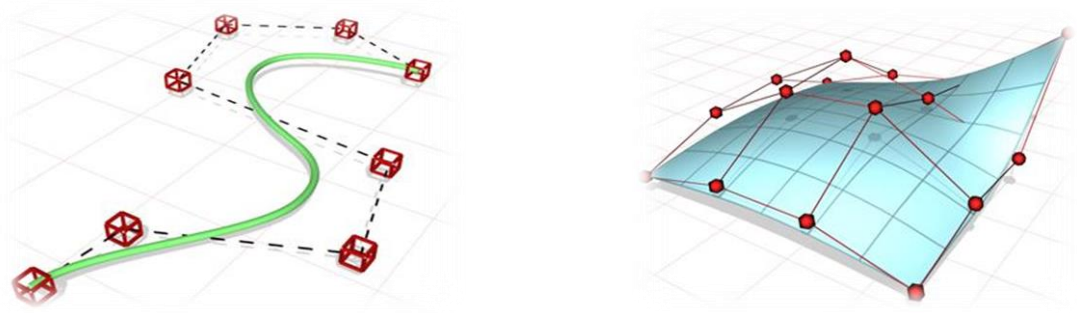
$$C(\xi) = \sum_{i=1}^{n} \left\{ R^{p}_{i}(\xi) \cdot X_i \right\}$$

2.11.6.2 Surfaces:

$$S(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \left\{ R^{p,q}_{i,j}(\xi,\eta) \cdot X_{i,j} \right\}$$

2.11.6.3 and Solids:

$$S(\xi,\eta,\zeta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} \left\{ R^{p,q,r}_{i,j,k}(\xi,\eta,\zeta) \cdot X_{i,j,k} \right\}$$



(a) **(b)**

Figure 2.24.
NURBS Geometries. **(a)** Curve, **(b)** Surfac

3. STIFFNESS MATRIX

3.1 PRELIMINARY STEPS FOR ANALYSIS

The main difference between isogeometric analysis and finite element analysis is the type of the shape functions used for the approximation of the solution field. The substitution of Lagrange polynomials with NURBS (widely utilized in the CAD community) leads to several new special characteristics.

3.1.1 Shape Functions

Finite element method shape functions are usually polynomials (e.g. Lagrange polynomials). However, they hold certain disadvantages. FEM shape functions are interpolatory at all nodes, internal and external, and have C^{-1} continuity at the edge. This results in the incompetence of FEM to define stresses and strains at any boundary of any element, leading in the introduction of other corrective methods such as extrapolation, in order to achieve that.

NURBS as shape functions hold an overlapping that is useful, as nearby elements are strongly connected, thus the simulation provides an improved approach to the natural problem. Continuous shape function derivatives lead to a continuous stress and strain field, minimizing the need for application of corrective methods.

3.1.2 Control Points

Classical FEM downsizes the natural problem of infinite unknowns to a finite number of unknowns, which are the degrees of freedom corresponding to the nodes. The position of the nodes depends on the element type. As a general rule, the nodes can be usually found in the corners and middle of the sides of the elements. They are part of the element and therefore part of the physical model. Displacements in other areas of the model can be approximated by a linear combination of displacements on the degrees of freedom. Distribution in the model is evaluated via the corresponding shape functions. In isoparametric elements, shape functions and their respective nodes are also used to approximate the geometry, thus enabling relatively complex shapes to be approximated with Lagrange polynomials.

In isogeometric analysis, NURBS are chosen as shape functions. The isoparametric concept is reversed, as geometrical mapping now defines the solution

approximation. The geometrical representation is achieved through a combination of control points and their corresponding shape functions. Degrees of freedom at the control points are now the unknowns.

3.1.3 Elements

Two ingredients of the isogeometric universe correspond to the essence of FEM "element". The isogeometric element could be either the patch or the knot span of the patch. In order to perform exact numerical integration, a certain number of Gauss points are chosen for the domain of every piece of the polynomial basis functions. The domain of this piece is the knot span, which resembles the finite element of FEM. This is the reason why in this thesis knot spans and not patches are considered isogeometric elements. In IGA, shape functions are not restricted to the interior of each element (knot span). Instead, they are non-zero across p+1 knot spans and overlap with more shape functions.

This overlapping results in a denser stiffness matrix than the classical finite element matrix even in the case of same number of dof. Apart from that, the fact that B-Spline functions are defined in the whole domain allows integration throughout the patch without building local element matrices separately. Of course, this would be time-consuming and it is not advised for advanced software technologies, but serves well for research purposes, where a flexible quadrature code is needed in order to test and discover new methods and ideas.

3.1.4 Gauss Points

3.1.1.1 Parametric Coordinates

As mentioned above, Gauss points are chosen for each knot span. Their coordinates are obtained on a reference element spanning [-1,1] as the roots of the Legendre polynomial. The next step is to transform the coordinates and weights from the reference knot span ξ^R to the desired knot span $[\xi_i, \xi_{i+1})$.

$$\xi = \frac{\left(\xi_{i+1} - \xi_i\right) \cdot \xi^R + \left(\xi_{i+1} + \xi_i\right)}{2} \qquad (3.1)$$

$$w^{GP\xi} = \frac{\left(\xi_{i+1} - \xi_i\right)}{2} \cdot w_\xi^R \qquad (3.2)$$

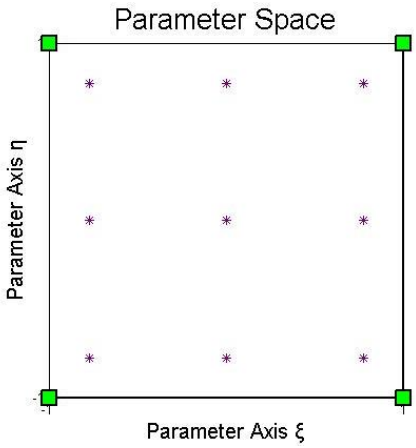
Full tensor product properties apply here as well, leading in similar equations for the other two parametric directions:

$$\eta = \frac{\left(\eta_{j+1} - \eta_j\right) \cdot \eta^R + \left(\eta_{j+1} + \eta_j\right)}{2} \qquad (3.3)$$

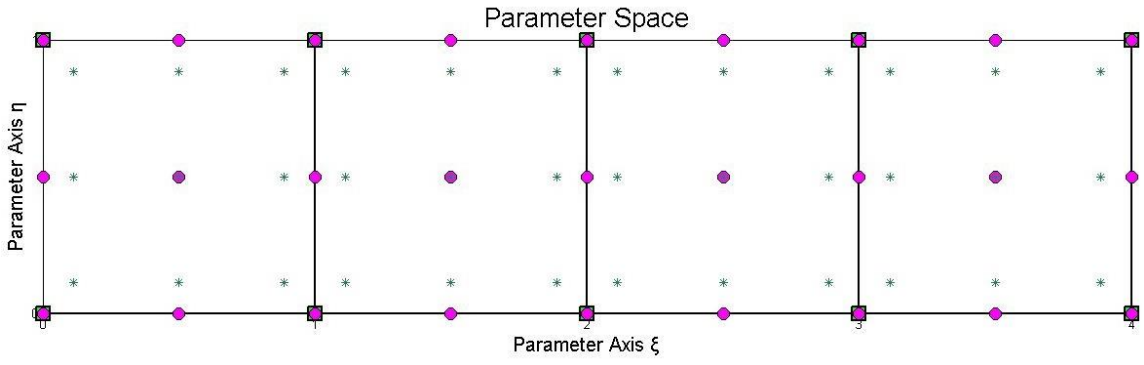
$$\zeta = \frac{\left(\zeta_{k+1} - \zeta_k\right) \cdot \zeta^R + \left(\zeta_{k+1} + \zeta_k\right)}{2} \qquad (3.4)$$

$$w^{GP\eta} = \frac{\left(\eta_{j+1} - \eta_j\right)}{2} \cdot w_\eta^R \qquad (3.5)$$

$$w^{GP\zeta} = \frac{\left(\zeta_{k+1} - \zeta_k\right)}{2} \cdot w_\zeta^R \qquad (3.6)$$



(a)



(b)

Figure 3.1. Gauss points
(a) on reference knot span and **(b)** transferred to parameter space.

3.1.1.2 Gauss Point Number

Gauss point coordinates and weights are evaluated for every patch. For the exact integration of a polynomial of degree p, (p+1)/2 or (p+2)/2 Gauss points are required per knot span for p being odd or even respectively.

For 1D problems, the maximum degree of the deformation matrix is defined from the derivation of piecewise polynomial shape functions and is p-1.

$$\left[\mathrm{B}(\xi)\right]^{\mathrm{T}}\cdot\left[\mathrm{E}\right]\cdot\left[\mathrm{B}(\xi)\right]$$

The aforementioned expression yields to the product of polynomials of maximum order p-1 resulting in a polynomial of maximum order (p-1)+(p-1=2p-2. Thus, the minimum number of Gauss points required per knot span for exact integration is:

$$\frac{(2\mathrm{p}-2)+2}{2}=\mathrm{p} \tag{3.7}$$

For 2D and 3D problems, the maximum order of the deformation matrix is determined by partial derivation of piecewise polynomial shape functions. Therefore the maximum degree is p for derivation in the remaining directions.

The order of the product $\left[\mathrm{B}(\xi)\right]^{\mathrm{T}}\cdot\left[\mathrm{E}\right]\cdot\left[\mathrm{B}(\xi)\right]$ is p+p=2p. In order to achieve exact integration, the minimum number of Gauss points per knot span is:

$$\frac{2\mathrm{p}+2}{2}=\mathrm{p}+1 \tag{3.8}$$

Conclusively per knot span:

- For 1D problems, **p** Gauss points per knot span are required.
- For 2D and 3D problems, **p+1** Gauss points per knot span are required.

3.1.5 Patches

As mentioned before, patches are used when a change in geometry type or material occurs. Patches can also be used in any case C^{-1} or C^{0} continuity is required. If separate knot vectors are used for every patch, stiffness matrices will be produced for every patch. The separate matrices are combined into one via connectivity arrays. If the connection is watertight, the procedure is the same as the one used in classical FEM to combine local element matrices to a single, global stiffness matrix.

3.1.6 Elasticity Matrix

Elasticity matrix types vary depending on the stress and strain field for each case. Data is obtained straight from classical FEM applications. Elasticity matrices for 1D elasticity, plane strain, plane stress and 3D elasticity are presented as follows, where E is the Young's modulus and ν is the Poisson's ratio.

1D elasticity:

$$\underset{(1x1)}{[E]} = E \tag{3.9}$$

2D elasticity, plane stress:

$$\underset{(3x3)}{[E]} = \frac{E}{1-\nu^2} \cdot \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{3.10}$$

2D elasticity, plane strain:

$$\underset{(3x3)}{[E]} = \frac{E}{(1-\nu)\cdot(1-2\nu)} \cdot \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{3.11}$$

3D elasticity:

$$\underset{(6x6)}{[E]} = \frac{E}{(1-\nu)\cdot(1-2\nu)} \cdot \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{3.12}$$

The corresponding stress and strain vectors are:

1D elasticity:

$$\underset{(1\times1)}{\{\sigma\}} = \{\sigma_X\} \tag{3.13}$$

$$\underset{(1\times1)}{\{\varepsilon\}} = \{\varepsilon_X\} \tag{3.14}$$

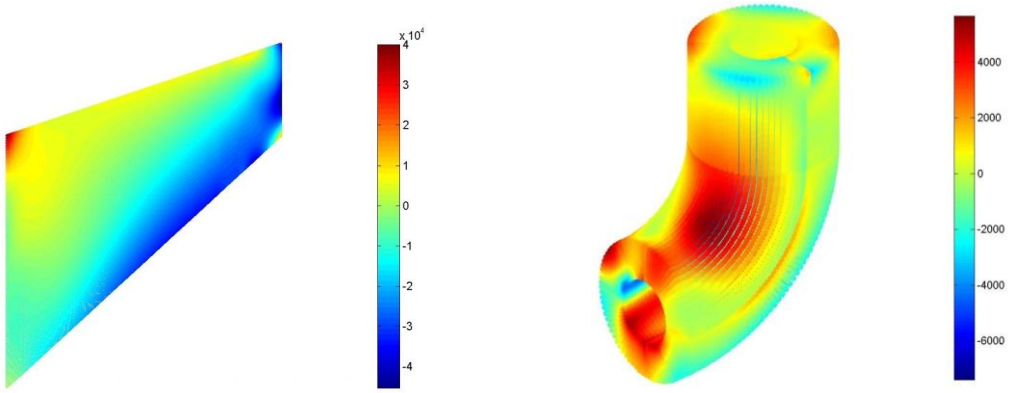
2D elasticity, plane stress and plane strain:

$$\underset{(3\times1)}{\{\sigma\}} = \begin{Bmatrix} \sigma_X \\ \sigma_Y \\ \tau_{XY} \end{Bmatrix} \tag{3.15}$$

$$\underset{(3\times1)}{\{\varepsilon\}} = \begin{Bmatrix} \varepsilon_X \\ \varepsilon_Y \\ \gamma_{XY} \end{Bmatrix} \tag{3.16}$$

3D elasticity:

$$\underset{(6\times1)}{\{\sigma\}} = \begin{bmatrix} \sigma_X \\ \sigma_Y \\ \sigma_Z \\ \sigma_{XY} \\ \sigma_{YZ} \\ \sigma_{ZX} \end{bmatrix}, \underset{(6\times1)}{\{\varepsilon\}} = \begin{bmatrix} \varepsilon_X \\ \varepsilon_Y \\ \varepsilon_Z \\ \gamma_{XY} \\ \gamma_{YZ} \\ \gamma_{ZX} \end{bmatrix} \tag{3.17}$$



(a) **(b)**

Figure 3.2.

Stress contour distribution. **(a)** σ_{xx} for plane stress, **(b)** τ_{yz} 3D elasticity

3.2 STIFFNESS MATRIX ASSEMBLY.

3.2.1 General Procedure

The general process for the global stiffness matrix assembly, as obtained from finite element method, is shown in the following flow chart:

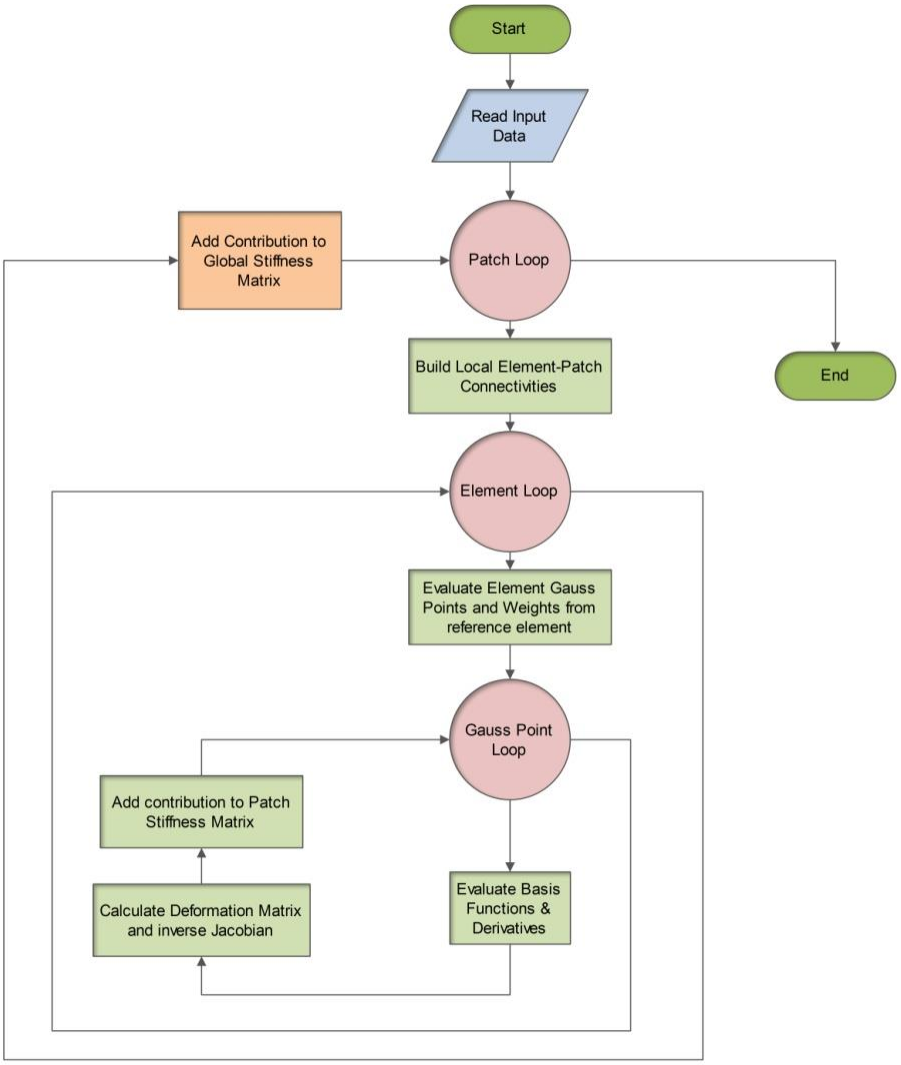


Figure 3.3.
Stiffness matrix assembly in finite element method.

There are three loops:

- Patch loop
- Element loop
- Gauss point loop

It is worth mentioning that the element loop in IGA can be avoided. In this case, the stiffness contribution of each control point pair is added directly to the stiffness

matrix of the patch. The reason for this is that parameter space is local to patch rather than elements.

Therefore, an engineer accustomed to the methods of isogeometric analysis knows that the element loop can be completely avoided, leading to the following flow chart:

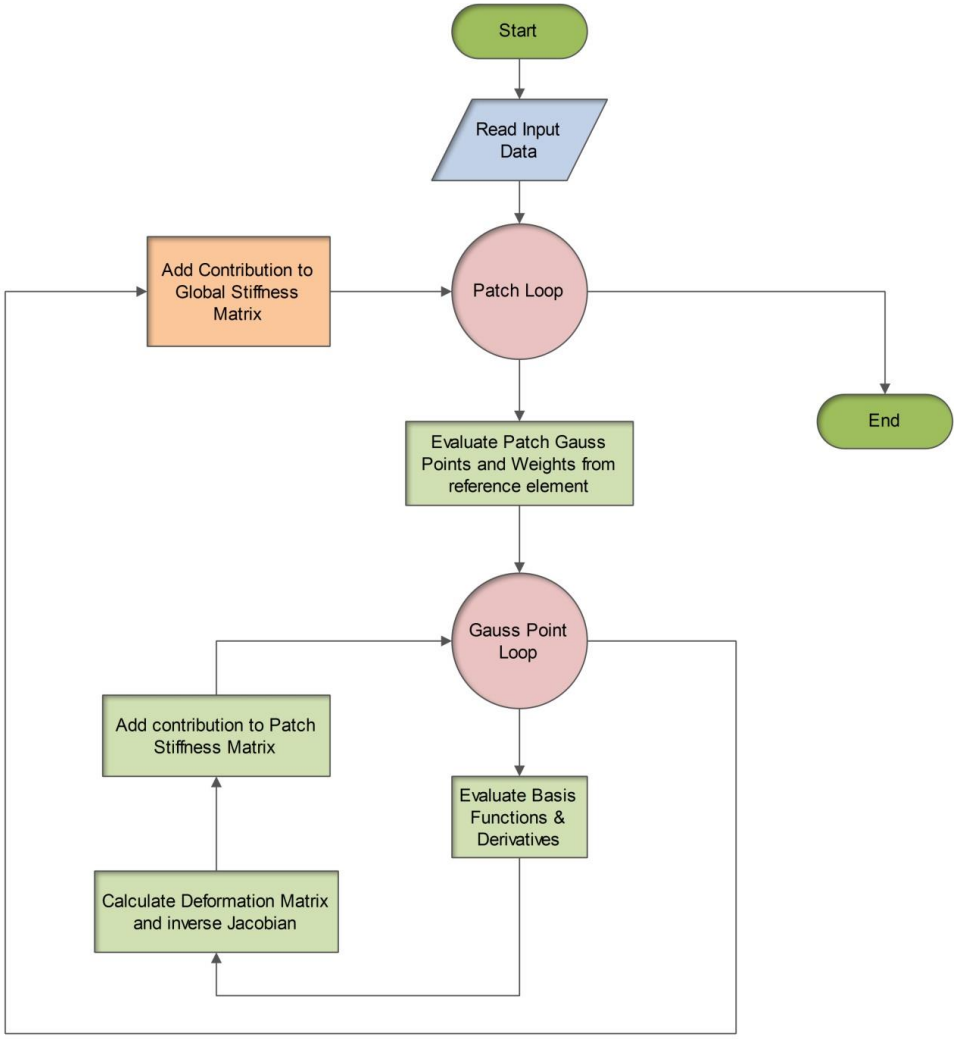


Figure 3.4.
Stiffness matrix assembly in isogeometric analysis

3.2.1.1 Input Data

Information necessary for the whole process of analysis is given as input at this point. The information essential for the formulation of the stiffness matrix is divided into two categories:

- Structural analysis and material
- Computational geometry

3.3 STIFFNESS MATRIX

3.3.1 Stiffness Matrix 1D

Let us consider a simple 1D application as an example. These applications are utilized only in truss systems with axial tension, but their significance is mostly academic. Simplifying the problem, without loss of generality, can lead to a better understanding of the principles involved. In our research career, we often address 1D analogies for the solution of complex problems and it has always been extremely helpful.

In such a case, there exists only axial deformation for each point of the truss. This deformation is u(x) = u(C(ξ)) = u(ξ). The respective strain matrix consists of one value:

$$\underset{(1\times1)}{\{\varepsilon\}} = [\varepsilon_x] = \left[\frac{\partial u}{\partial x}\right] \tag{3.18}$$

In order to calculate the derivative of u, we must first establish a transition from physical space to parameter space:

$$\frac{\partial \varphi}{\partial x} = \frac{\partial \varphi}{\partial \xi}\frac{\partial \xi}{\partial x} \tag{3.19}$$

$$\frac{\partial \varphi}{\partial \xi} = \frac{\partial \varphi}{\partial x}\frac{\partial x}{\partial \xi} \tag{3.20}$$

$$\left[\frac{\partial \varphi}{\partial \xi}\right] = [J]\cdot\left[\frac{\partial \varphi}{\partial x}\right] \tag{3.21}$$

where [J] is the Jacobian matrix enabling transition from physical to parameter space and vice-versa. It can be evaluated with the help of basis functions $R_i(\xi)$ and the control points' Cartesian coordinates X_i as shown:

$$\underset{(1\times1)}{\left[J(\xi)\right]} = \underset{(1\times n)}{\left[R_{1,\xi}(\xi) \quad R_{2,\xi}(\xi) \quad \dots \quad \dots \quad \dots \quad R_{n,\xi}(\xi)\right]}\cdot\underset{(n\times1)}{\begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_n \end{bmatrix}} \tag{3.22}$$

where $R_{i,\xi}(\xi) = \frac{\partial}{\partial \xi}R_i(\xi)$.

In finite element analysis, the reverse transformation is utilized. This is why the inverse matrix $[J]^{-1}$ is needed.

$$\underset{(1x1)}{\left[J\right]^{-1}} = \frac{1}{\underset{(1\times1)}{\left[J\right]}} \tag{3.23}$$

Special care has to be taken in order to correctly calculate the Jacobian matrix. The positive direction of the axes in parameter and physical space must coincide, or the determinant of the Jacobian will be negative and the matrix [J] irreversible. Numerical integration on points of singularity, such as two points on parameter space mapped into the same point on physical space, has to be avoided as well.

The next step is to calculate the matrices $[B_1]$ and $[B_2]$. The matrix $[B_1]$ transfers the strains of the element from parameter to physical space and the matrix $[B_2]$ transfers the nodal displacements of the elements to the strains at the parameter space. Therefore, the matrices $[B_1]$ and $[B_2]$ can be calculated from the equations below:

$$\underset{(1\times1)}{\left\{\varepsilon\right\}} = \left[\frac{\partial u}{\partial x}\right] = \left[J\right]^{-1} \cdot \left[\frac{\partial u}{\partial \xi}\right] \tag{3.24}$$

$$\underset{(1\times1)}{\left[B_1(\xi)\right]} = \left[\frac{1}{J_{11}}\right] \tag{3.25}$$

$$\underset{(1\times1)}{\left\{\varepsilon\right\}} = \underset{(1\times1)}{\left[B_1\right]} \cdot \underset{(1\times1)}{\left\{u_\xi\right\}} \tag{3.26}$$

$$\left[\frac{\partial u}{\partial \xi}\right] = \left[R_{1,\xi}(\xi) \quad R_{2,\xi}(\xi) \quad \dots \quad \dots \quad \dots \quad R_{n,\xi}(\xi)\right] \cdot \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_n \end{bmatrix} \tag{3.27}$$

$$\underset{(1\times n)}{\left[B_2(\xi)\right]} = \left[R_{1,\xi}(\xi) \quad R_{2,\xi}(\xi) \quad \dots \quad \dots \quad \dots \quad R_{n,\xi}(\xi)\right] \tag{3.28}$$

$$\underset{(1\times1)}{\left\{u\right\}} = \underset{(1\times n)}{\left[B_2\right]} \cdot \underset{(n\times1)}{\left\{d\right\}} \tag{3.29}$$

After that, the Deformation matrix is evaluated.

$$\underset{(1\times n)}{\left[B(\xi)\right]} = \underset{(1\times1)}{\left[B_1(\xi)\right]} \cdot \underset{(1\times n)}{\left[B_2(\xi)\right]}$$

Deformation matrix produces strain values anywhere in the model, by utilizing nodal displacements.

$$\underset{(1\times1)}{\{\varepsilon(\xi)\}} = \underset{(1\times n)}{[B(\xi)]} \cdot \underset{(n\times1)}{\{d\}} \tag{3.30}$$

The stiffness matrix for the patch is evaluated as shown:

$$\underset{(n\times n)}{[K]} = \int_{\xi_0}^{\xi_{n+p+1}} \underset{(n\times1)}{[B(\xi)]^T} \cdot \underset{(1x1)}{[E]} \cdot \underset{(1\times n)}{[B(\xi)]} \cdot A \cdot \det[J]\, d\xi \tag{3.31}$$

Direct integration is almost never applicable. Numerical integration is used instead, looping through all the Gauss points of a patch and their respective weights:

$$\underset{(n\times n)}{[K]} = \sum_{i=1}^{GP\xi} \left\{ \underset{(n\times1)}{[B(\xi_i)]^T} \cdot \underset{(1x1)}{[E]} \cdot \underset{(1\times n)}{[B(\xi_i)]} \cdot A \cdot \det[J] \cdot w_i^{GP\xi} \right\} \tag{3.32}$$

where

- A : the area of the cross-section
- GP_ξ : the total number of Gauss points for the specific patch
- ξ_i : the coordinates of the Gauss points
- $w_i^{GP\xi}$: the corresponding weights.

3.3.2 Stiffness Matrix 2D

2D elasticity problems have many applications in modern analysis. The logic is exactly the same as in 1D problems. The main difference is, obviously, the utilization of one more dimension. Parameter space is defined on (ξ,η) and physical space on (x,y). Displacements per x, y at any point in the entire domain are defined as u(x,y)= u(S(ξ,η))= u(ξ,η) and v(x,y)= v(ξ,η) respectively.

The strain vector is defined as:

$$\underset{(3\times1)}{\{\varepsilon\}} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{bmatrix} \Rightarrow \underset{(3\times1)}{\{\varepsilon\}} = \begin{bmatrix} \dfrac{\partial}{\partial x} & 0 \\ 0 & \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x} \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.33}$$

The transformation of a function ϕ between parameter and physical space yields:

$$\frac{\partial \varphi}{\partial x} = \frac{\partial \varphi}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \varphi}{\partial \eta}\frac{\partial \eta}{\partial x} \tag{3.34}$$

$$\frac{\partial \varphi}{\partial y} = \frac{\partial \varphi}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial \varphi}{\partial \eta}\frac{\partial \eta}{\partial y} \tag{3.35}$$

$$\frac{\partial \varphi}{\partial \xi} = \frac{\partial \varphi}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \varphi}{\partial y}\frac{\partial y}{\partial \xi} \tag{3.36}$$

$$\frac{\partial \varphi}{\partial \eta} = \frac{\partial \varphi}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \varphi}{\partial y}\frac{\partial y}{\partial \eta} \tag{3.37}$$

Thus, the 2D Jacobian matrix can be defined as:

$$\begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix} = \underset{(2\times 2)}{[J]} \cdot \begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix} \tag{3.38}$$

and the inverse mapping:

$$\begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix} = \underset{(2\times 2)}{[J]}^{-1} \cdot \begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix} \tag{3.39}$$

The Jacobian matrix can be evaluated as shown:

$$\underset{(2\times 2)}{[J]} = \begin{bmatrix} R_{1,\xi}(\xi,\eta) & R_{2,\xi}(\xi,\eta) & \dots & \dots & \dots & R_{N,\xi}(\xi,\eta) \\ R_{1,\eta}(\xi,\eta) & R_{2.\eta}(\xi,\eta) & \dots & \dots & \dots & R_{N,\eta}(\xi,\eta) \end{bmatrix} \cdot \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ . & . \\ . & . \\ . & . \\ X_N & Y_N \end{bmatrix} \tag{3.40}$$

where N= n x m is the total number of control points.

The inverse Jacobian matrix is used in stiffness matrix calculation:

$$\underset{(2\times2)}{\left[J\right]^{-1}}=\begin{bmatrix} J^{*}_{11} & J^{*}_{12} \\ J^{*}_{21} & J^{*}_{22} \end{bmatrix}=\frac{1}{\det\left[J\right]}\cdot\begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \tag{3.41}$$

The determinant of the Jacobian matrix is also required and is equal to:

$$\det\left[J\right]=J_{11}\cdot J_{22}-J_{21}\cdot J_{12} \tag{3.42}$$

In order to calculate the deformation matrix for 2D problems, $[B_1]$ and $[B_2]$ have to be evaluated as usual.

To obtain matrix $[B_1]$:

$$\underset{(3\times1)}{\left\{\varepsilon\right\}}=\begin{bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y}+\dfrac{\partial v}{\partial x} \end{bmatrix}=\frac{1}{\det\left[J\right]}\cdot\begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix}\cdot\begin{bmatrix} \dfrac{\partial u}{\partial \xi} \\ \dfrac{\partial u}{\partial \eta} \\ \dfrac{\partial v}{\partial \xi} \\ \dfrac{\partial v}{\partial \eta} \end{bmatrix} \tag{3.43}$$

Hence,

$$\underset{(3\times4)}{\left[B_1(\xi,\eta)\right]}=\frac{1}{\det\left[J\right]}\cdot\begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix} \tag{3.44}$$

To calculate matrix $[B_2]$:

$$\begin{bmatrix} \dfrac{\partial u}{\partial \xi} \\ \dfrac{\partial u}{\partial \eta} \\ \dfrac{\partial v}{\partial \xi} \\ \dfrac{\partial v}{\partial \eta} \end{bmatrix}=\begin{bmatrix} R_{1,\xi} & 0 & R_{2,\xi} & 0 & \dots & \dots & \dots & R_{N,\xi} & 0 \\ R_{1,\eta} & 0 & R_{2,\eta} & 0 & \dots & \dots & \dots & R_{N,\eta} & 0 \\ 0 & R_{1,\xi} & 0 & R_{2,\xi} & \dots & \dots & \dots & 0 & R_{N,\xi} \\ 0 & R_{1,\eta} & 0 & R_{2,\eta} & \dots & \dots & \dots & 0 & R_{N,\eta} \end{bmatrix}\cdot\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ . \\ . \\ . \\ u_N \\ v_N \end{bmatrix} \tag{3.45}$$

Hence,

$$\underset{(4\times 2N)}{\left[B_2(\xi,\eta)\right]}=\begin{bmatrix} R_{1,\xi} & 0 & R_{2,\xi} & 0 & \dots & \dots & \dots & R_{N,\xi} & 0 \\ R_{1,\eta} & 0 & R_{2,\eta} & 0 & \dots & \dots & \dots & R_{N,\eta} & 0 \\ 0 & R_{1,\xi} & 0 & R_{2,\xi} & \dots & \dots & \dots & 0 & R_{N,\xi} \\ 0 & R_{1,\eta} & 0 & R_{2,\eta} & \dots & \dots & \dots & 0 & R_{N,\eta} \end{bmatrix} \tag{3.46}$$

Having determined $[B_1]$ and $[B_2]$, the deformation matrix is calculated as:

$$\underset{(3\times 2N)}{\left[B(\xi,\eta)\right]}=\underset{(3\times 4)}{\left[B_1(\xi,\eta)\right]}\cdot\underset{(4\times 2N)}{\left[B_2(\xi,\eta)\right]} \tag{3.47}$$

In order to evaluate the stiffness matrix, integration is required.

$$\underset{(2N\times 2N)}{[K]}=\int_{\xi_0}^{\xi_{n+p+1}}\int_{\eta_0}^{\eta_{m+q+1}}\underset{(2N\times 3)}{\left[B(\xi,\eta)\right]^T}\cdot\underset{(3x3)}{[E]}\cdot\underset{(3\times 2N)}{\left[B(\xi,\eta)\right]}\cdot t\cdot\det[J]\,d\eta d\xi \tag{3.48}$$

Numerical integration procedures for ξ,η lead to integration of tensor product Gauss points.

$$\underset{(2N\times 2N)}{[K]}=\sum_{i=1}^{GP\xi}\sum_{j=1}^{GP\eta}\underset{(2N\times 3)}{\left[B(\xi_i,\eta_j)\right]^T}\cdot\underset{(3x3)}{[E]}\cdot\underset{(3\times 2N)}{\left[B(\xi_i,\eta_j)\right]}\cdot t\cdot\det[J]\cdot w_i^{GP\xi}\cdot w_j^{GP\eta} \tag{3.49}$$

where:

- t: the thickness of the cross-section
- GP_ξ: the total number of Gauss points per ξ for the specific patch
- GP_η: the total number of Gauss points per η for the specific patch
- ξ_i, η_j: the coordinates of the tensor product Gauss point i, j
- $w_i^{GP\xi}$, $w_j^{GP\eta}$: the corresponding weights

The only difference, at this point, between plane stress and plane strain is the elasticity matrix which is the result of the utilized constitutive law.

3.3.3 Stiffness Matrix 3D

3D elasticity is merely the extension of 2D elasticity in all directions, with a complete stress field. Every other problem can be created by downgrading 3D problems into 2D and 1D problems.

The displacement field for each point in physical space is now defined for x, y, z by u(x,y,z)= u(S(ξ,η,ζ))= u(ξ,η,ζ), v(ξ,η,ζ), w(ξ,η,ζ) respectively. The strain field can now be defined as:

$$\underset{(6\times1)}{\{\varepsilon\}}=\begin{bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{bmatrix}=\begin{bmatrix}\frac{\partial u}{\partial x}\\ \frac{\partial v}{\partial y}\\ \frac{\partial w}{\partial z}\\ \frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\\ \frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\\ \frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\end{bmatrix}=\begin{bmatrix}\frac{\partial}{\partial x} & 0 & 0\\ 0 & \frac{\partial}{\partial y} & 0\\ 0 & 0 & \frac{\partial}{\partial z}\\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0\\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y}\\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x}\end{bmatrix}\cdot\begin{bmatrix}u\\ v\\ w\end{bmatrix} \tag{3.50}$$

, which leads to the definition of the Jacobian matrix for 3D:

$$\begin{bmatrix}\frac{\partial\varphi}{\partial\xi}\\ \frac{\partial\varphi}{\partial\eta}\\ \frac{\partial\varphi}{\partial\zeta}\end{bmatrix}=\begin{bmatrix}\frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} & \frac{\partial z}{\partial\xi}\\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} & \frac{\partial z}{\partial\eta}\\ \frac{\partial x}{\partial\zeta} & \frac{\partial y}{\partial\zeta} & \frac{\partial z}{\partial\zeta}\end{bmatrix}\cdot\begin{bmatrix}\frac{\partial\varphi}{\partial x}\\ \frac{\partial\varphi}{\partial y}\\ \frac{\partial\varphi}{\partial z}\end{bmatrix}\Rightarrow\begin{bmatrix}\frac{\partial\varphi}{\partial\xi}\\ \frac{\partial\varphi}{\partial\eta}\\ \frac{\partial\varphi}{\partial\zeta}\end{bmatrix}=\underset{3\times3}{[J]}\cdot\begin{bmatrix}\frac{\partial\varphi}{\partial x}\\ \frac{\partial\varphi}{\partial y}\\ \frac{\partial\varphi}{\partial z}\end{bmatrix} \tag{3.51}$$

and the inverse Jacobian matrix as well:

$$\begin{bmatrix}\frac{\partial\varphi}{\partial x}\\ \frac{\partial\varphi}{\partial y}\\ \frac{\partial\varphi}{\partial z}\end{bmatrix}=\underset{(3\times3)}{[J]^{-1}}\cdot\begin{bmatrix}\frac{\partial\varphi}{\partial\xi}\\ \frac{\partial\varphi}{\partial\eta}\\ \frac{\partial\varphi}{\partial\zeta}\end{bmatrix} \tag{3.52}$$

Jacobian matrix can be calculated from the derivatives of the shape functions.

$$\underset{(3\times3)}{[J]}=\begin{bmatrix}R_{1,\xi}(\xi,\eta,\zeta) & R_{2,\xi}(\xi,\eta,\zeta) & \dots & \dots & \dots & R_{N,\xi}(\xi,\eta,\zeta)\\ R_{1,\eta}(\xi,\eta,\zeta) & R_{2,\eta}(\xi,\eta,\zeta) & \dots & \dots & \dots & R_{N,\eta}(\xi,\eta,\zeta)\\ R_{1,\zeta}(\xi,\eta,\zeta) & R_{2,\zeta}(\xi,\eta,\zeta) & \dots & \dots & \dots & R_{N,\zeta}(\xi,\eta,\zeta)\end{bmatrix}\begin{bmatrix}X_1 & Y_1 & Z_1\\ X_2 & Y_2 & Z_2\\ . & . & .\\ . & . & .\\ . & . & .\\ X_N & Y_N & Z_N\end{bmatrix}$$

(3.53)

The inverse of the Jacobian matrix is:

$$\underset{(3\times3)}{\left[\mathrm{J}\right]}^{-1}=\begin{bmatrix}\mathrm{J}_{11}^{*} & \mathrm{J}_{12}^{*} & \mathrm{J}_{13}^{*}\\ \mathrm{J}_{21}^{*} & \mathrm{J}_{22}^{*} & \mathrm{J}_{23}^{*}\\ \mathrm{J}_{31}^{*} & \mathrm{J}_{32}^{*} & \mathrm{J}_{33}^{*}\end{bmatrix} \tag{3.54}$$

$[B_1]$ is evaluated as:

$$\underset{(6\times9)}{\left[\mathrm{B}_1\left(\xi,\eta,\zeta\right)\right]}=\begin{bmatrix}\mathrm{J}_{11}^{*} & \mathrm{J}_{12}^{*} & \mathrm{J}_{13}^{*} & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & \mathrm{J}_{21}^{*} & \mathrm{J}_{22}^{*} & \mathrm{J}_{23}^{*} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{J}_{31}^{*} & \mathrm{J}_{32}^{*} & \mathrm{J}_{33}^{*}\\ \mathrm{J}_{21}^{*} & \mathrm{J}_{22}^{*} & \mathrm{J}_{23}^{*} & \mathrm{J}_{11}^{*} & \mathrm{J}_{12}^{*} & \mathrm{J}_{13}^{*} & 0 & 0 & 0\\ 0 & 0 & 0 & \mathrm{J}_{31}^{*} & \mathrm{J}_{32}^{*} & \mathrm{J}_{33}^{*} & \mathrm{J}_{21}^{*} & \mathrm{J}_{22}^{*} & \mathrm{J}_{23}^{*}\\ \mathrm{J}_{31}^{*} & \mathrm{J}_{32}^{*} & \mathrm{J}_{33}^{*} & 0 & 0 & 0 & \mathrm{J}_{11}^{*} & \mathrm{J}_{12}^{*} & \mathrm{J}_{13}^{*}\end{bmatrix} \tag{3.55}$$

As for $[B_2]$:

$$\underset{(9\times3N)}{\left[\mathrm{B}_2\left(\xi,\eta,\zeta\right)\right]}=\begin{bmatrix}\mathrm{R}_{1,\xi} & 0 & 0 & \mathrm{R}_{2,\xi} & 0 & 0 & \dots & \dots & \dots & \mathrm{R}_{\mathrm{N},\xi} & 0 & 0\\ \mathrm{R}_{1,\eta} & 0 & 0 & \mathrm{R}_{2,\eta} & 0 & 0 & \dots & \dots & \dots & \mathrm{R}_{\mathrm{N},\eta} & 0 & 0\\ \mathrm{R}_{1,\zeta} & 0 & 0 & \mathrm{R}_{2,\zeta} & 0 & 0 & \dots & \dots & \dots & \mathrm{R}_{\mathrm{N},\zeta} & 0 & 0\\ 0 & \mathrm{R}_{1,\xi} & 0 & 0 & \mathrm{R}_{2,\xi} & 0 & \dots & \dots & \dots & 0 & \mathrm{R}_{\mathrm{N},\xi} & 0\\ 0 & \mathrm{R}_{1,\eta} & 0 & 0 & \mathrm{R}_{2,\eta} & 0 & \dots & \dots & \dots & 0 & \mathrm{R}_{\mathrm{N},\eta} & 0\\ 0 & \mathrm{R}_{1,\zeta} & 0 & 0 & \mathrm{R}_{2,\zeta} & 0 & \dots & \dots & \dots & 0 & \mathrm{R}_{\mathrm{N},\zeta} & 0\\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ 0 & 0 & \mathrm{R}_{1,\xi} & 0 & 0 & \mathrm{R}_{2,\xi} & \dots & \dots & \dots & 0 & 0 & \mathrm{R}_{\mathrm{N},\xi}\\ 0 & 0 & \mathrm{R}_{1,\eta} & 0 & 0 & \mathrm{R}_{2,\eta} & \dots & \dots & \dots & 0 & 0 & \mathrm{R}_{\mathrm{N},\eta}\\ 0 & 0 & \mathrm{R}_{1,\zeta} & 0 & 0 & \mathrm{R}_{2,\zeta} & \dots & \dots & \dots & 0 & 0 & \mathrm{R}_{\mathrm{N},\zeta}\end{bmatrix} \tag{3.56}$$

As a result, the deformation matrix for 3D elasticity is calculated as:

$$\underset{(6\times3N)}{\left[\mathrm{B}\left(\xi,\eta,\zeta\right)\right]}=\underset{(6\times9)}{\left[\mathrm{B}_1\left(\xi,\eta,\zeta\right)\right]}\cdot\underset{(9\times3N)}{\left[\mathrm{B}_2\left(\xi,\eta,\zeta\right)\right]} \tag{3.57}$$

The corresponding stiffness matrix is produced by integration

$$\underset{(3N\times3N)}{\left[\mathrm{K}\right]}=\int_{\xi_0}^{\xi_{n+p+1}}\int_{\eta_0}^{\eta_{m+q+1}}\int_{\zeta_0}^{\zeta_{l+r+1}}\underset{(3N\times6)}{\left[\mathrm{B}\left(\xi,\eta,\zeta\right)\right]^{\mathrm{T}}}\cdot\underset{(6x6)}{\left[\mathrm{E}\right]}\cdot\underset{(6\times3N)}{\left[\mathrm{B}\left(\xi,\eta,\zeta\right)\right]}\cdot\det\left[\mathrm{J}\right]\,\mathrm{d}\zeta\mathrm{d}\eta\mathrm{d}\xi \tag{3.58}$$

Numerical integration is used in 3D as well

$$\underset{(3N\times 3N)}{[K]} = \sum_{i=1}^{GP\xi}\sum_{j=1}^{GP\eta}\sum_{k=1}^{GP\zeta}\underset{(3N\times 6)}{\left[B\left(\xi_i,\eta_j,\zeta_k\right)\right]^T}\cdot\underset{(6x6)}{[E]}\cdot\underset{(6\times 3N)}{\left[B\left(\xi_i,\eta_j,\zeta_k\right)\right]}\cdot\det[J]\cdot w_i^{GP\xi}\cdot w_j^{GP\eta}\cdot w_k^{GP\zeta} \tag{3.59}$$

where:

- GP_{ξ} : the total number of Gauss points per ξ for the specific patch
- GP_{η} : the total number of Gauss points per η for the specific patch
- GP_{ζ} : the total number of Gauss points per ζ for the specific patch
- ξ_i, η_j, ζ_k : the coordinates of the tensor product Gauss point
- $w_i^{GP\xi}$, $w_j^{GP\eta}$, $w_k^{GP\zeta}$: the corresponding weights of Gauss points

3.4 SOLVING STIFFNESS MATRIX

3.4.1 The Cholesky Method

3.4.1.1 Classical Method

Cholesky method is a variation of the classical Gauss elimination method. This method is used in order to compute displacements for the external load vectors which are not known by the time of the initial solution. These cases are usually met in non-linear analysis and dynamic analysis of structures.

This technique, factorizes the stiffness matrix as following:

$$[K]=[L]\cdot[L]^T \tag{3.60}$$

where [*L*] is a lower triangular matrix. This method is applied to symmetrical and positive matrices.

3.4.1.2 Variation for non-symmetrical matrices

A variation of this method is used mostly to non-symmetrical matrices and has the form:

$$[K]=[L]\cdot[U] \tag{3.61}$$

where [*L*], [*U*] are one upper and one lower triangular matrix respectively.

3.4.1.3 Crout Variation

The Crout variation of the classical Cholesky method, is mainly used for non-linear analysis of structures. Although this method is slightly different from the classical one, it is generally more stable and dependent. Thus, it is used in a wide variety of problems by the computational mechanics society.

The stiffness matrix can be expressed now as:

$$[K]=[L]\cdot[Q][L]^T \tag{3.62}$$

where [*L*] is a lower triangular matrix with the value 1 over its diagonal and [*Q*] is just a diagonal matrix.

After the factorization of the stiffness matrix, solving the matrix equations is expresses algebraically as:

$$[D]=[L]^{-T}[Q]^{-1}[L]^{-1}\{R\} \tag{3.63}$$

and it is performed in two steps:

Step 1 Solution of the subproblem

$$\begin{aligned}&[L]\{x\}=\{R\}\Rightarrow\\&\{x\}=[L]^{-1}\{R\}\end{aligned} \tag{3.64}$$

Step 2 Solution of the subproblem

$$[Q][L]^T\{D\}=\{x\} \tag{3.65}$$

$$\{y\}=[Q]^{-1}\{x\} \tag{3.66}$$

$$\{D\}=[L]^{-T}\{y\} \tag{3.67}$$

The calculation of [*L*] and [*Q*] values is arising from the solution of equations presented in the relation (3.68)

$$[K]=[L][Q][L]^T\Rightarrow$$

$$[K]=\begin{bmatrix}I & 0 & 0\\ L_{21} & I & 0\\ L_{31} & L_{32} & I\end{bmatrix}\begin{bmatrix}Q_1 & 0 & 0\\ 0 & Q_2 & 0\\ 0 & 0 & Q_3\end{bmatrix}\begin{bmatrix}I & L_{21}^T & L_{31}^T\\ 0 & I & L_{32}^T\\ 0 & 0 & I\end{bmatrix} \tag{3.6}$$

4. EXTERNAL LOADS, BOUNDARY CONDITIONS AND STRESS FIELD

4.1 ANALYSIS

4.1.1 External Load

After the stiffness matrix assembly, the external load vector has to be calculated. Concentrated loads can be assigned directly at the degrees of freedom. Distributed loads f(ξ,η,ζ) have to be transformed into equivalent concentrated loads by integration:

$$\underset{(N\times 1)}{\{F\}} = \int_{\xi_0}^{\xi_{n+p+1}} \int_{\eta_0}^{\eta_{m+q+1}} \int_{\zeta_0}^{\zeta_{l+r+1}} \underset{(N\times 1)}{\{R(\xi,\eta,\zeta)\}} \cdot \underset{(1\times 1)}{f(\xi,\eta,\zeta)} \cdot \det[J]\, d\zeta d\eta d\xi \tag{4.1}$$

More specifically, for each case:

1D:

$$\underset{(N\times 1)}{\{F\}} = \int_{\xi_0}^{\xi_{n+p+1}} \underset{(N\times 1)}{\{R(\xi)\}} \cdot \underset{(1\times 1)}{f(\xi)} \cdot \det[J] d\xi \tag{4.2}$$

2D:

$$\underset{(N\times 1)}{\{F\}} = \int_{\xi_0}^{\xi_{n+p+1}} \int_{\eta_0}^{\eta_{m+q+1}} \underset{(N\times 1)}{\{R(\xi,\eta)\}} \cdot \underset{(1\times 1)}{f(\xi,\eta)} \cdot \det[J] d\eta d\xi \tag{4.3}$$

3D:

$$\underset{(N\times 1)}{\{F\}} = \int_{\xi_0}^{\xi_{n+p+1}} \int_{\eta_0}^{\eta_{m+q+1}} \int_{\zeta_0}^{\zeta_{l+r+1}} \underset{(N\times 1)}{\{R(\xi,\eta,\zeta)\}} \cdot \underset{(1\times 1)}{f(\xi,\eta,\zeta)} \cdot \det[J] d\zeta d\eta d\xi \tag{4.4}$$

This way, the load vector $\{F\}$ is assembled.

4.1.2 Refined Load

If a load vector has already been evaluated for a coarse mesh, there is no need to apply numerical integration for the new load vector of the fine mesh. New load values can be evaluated directly from the coarse mesh.

Let $\{L^C\}$, $\{L^F\}$ be the load vectors, $\{N^C(\xi)\},\{N^F(\xi)\}$ the basis functions and $\{X^C\}$, $\{X^F\}$ the control point Cartesian coordinates for the coarse and fine mesh respectively. Let $f(\xi)$ be the load distribution in the physical space. It applies that:

$$\underset{(n\times1)}{\left\{L^C\right\}} = \int_{\xi_0}^{\xi_{max}} \underset{(n\times1)}{\left\{N^C(\xi)\right\}} \cdot \underset{(1\times1)}{f(\xi)}\, d\xi \tag{4.5}$$

Both meshes provide the same geometrical representation, resulting in:

$$\underset{(1\times n)}{C(\xi)} = \underset{(1\times n)}{\left\{N^C(\xi)\right\}^T} \cdot \underset{(n\times1)}{\left\{X^C\right\}} \tag{4.6}$$

$$\underset{(1\times m)}{C(\xi)} = \underset{(1\times m)}{\left\{N^F(\xi)\right\}^T} \cdot \underset{(m\times1)}{\left\{X^F\right\}} \tag{4.7}$$

Thus,

$$\underset{(1\times m)}{\left\{N^F(\xi)\right\}^T} \cdot \underset{(m\times1)}{\left\{X^F\right\}} = \underset{(1\times n)}{\left\{N^C(\xi)\right\}^T} \cdot \underset{(n\times1)}{\left\{X^C\right\}} \tag{4.8}$$

It has been established that:

$$\underset{(n\times1)}{\left\{X^C\right\}} = \underset{(n\times m)}{\left[T^{CF}\right]} \cdot \underset{(m\times1)}{\left\{X^F\right\}} \tag{4.9}$$

Therefore,

$$\underset{(1\times m)}{\left\{N^F(\xi)\right\}^T} \cdot \underset{(m\times1)}{\left\{X^F\right\}} = \underset{(1\times n)}{\left\{N^C(\xi)\right\}^T} \cdot \underset{(n\times1)}{\left\{X^C\right\}} = \underset{(1\times n)}{\left\{N^C(\xi)\right\}^T} \cdot \underset{(n\times m)}{\left[T^{CF}\right]} \cdot \underset{(m\times1)}{\left\{X^F\right\}} \Rightarrow$$

$$\underset{(1\times m)}{\left\{N^F(\xi)\right\}^T} = \underset{(1\times n)}{\left\{N^C(\xi)\right\}^T} \cdot \underset{(n\times m)}{\left[T^{CF}\right]} \tag{4.10}$$

This leads to

$$\underset{m\times1}{\left\{N^F(\xi)\right\}} = \underset{m\times n}{\left[T^{CF}\right]^T} \cdot \underset{n\times1}{\left\{N^C(\xi)\right\}} \tag{4.11}$$

The fine mesh load vector is evaluated by:

$$\underset{(m\times 1)}{\left\{L^F\right\}} = \int_{\xi_0}^{\xi_{max}} \underset{(m\times 1)}{\left\{N^F(\xi)\right\}} \cdot \underset{(1\times 1)}{f(\xi)}\, d\xi \Rightarrow$$

$$\underset{(m\times 1)}{\left\{L^F\right\}} = \int_{\xi_0}^{\xi_{max}} \underset{(m\times n)}{\left[T^{CF}\right]^T} \cdot \underset{(n\times 1)}{\left\{N^C(\xi)\right\}} \cdot \underset{(1\times 1)}{f(\xi)}\, d\xi \Rightarrow \tag{4.12}$$

$$\underset{(m\times 1)}{\left\{L^F\right\}} = \underset{(m\times n)}{\left[T^{CF}\right]^T} \int_{\xi_0}^{\xi_{max}} \underset{(n\times 1)}{\left\{N^C(\xi)\right\}} \cdot \underset{(1\times 1)}{f(\xi)}\, d\xi$$

In conclusion,

$$\underset{(m\times 1)}{\left\{L^F\right\}} = \underset{(m\times n)}{\left[T^{CF}\right]^T} \cdot \underset{(n\times 1)}{\left\{L^C\right\}} \tag{4.13}$$

In the same manner, equivalent loads for reverse-refinement can be defined as:

$$\underset{(n\times 1)}{\left\{L^C\right\}} = \underset{(n\times m)}{\left[T^{FC}\right]^T} \cdot \underset{(m\times 1)}{\left\{L^F\right\}} \tag{4.14}$$

An engineer must always bear in mind the restrictions set for reverse refinement. Both in geometrical representation and in equivalent load evaluation, this procedure only works under certain circumstances. More specifically, loads refined from a coarse mesh can be transformed back to that coarse mesh. On the other hand, loads created initially from a fine mesh will probably be transformed incorrectly to the coarse mesh.

4.1.3 Boundary Conditions

Certain degrees of freedom are fixed, in that their displacements are zero. These are called stationary and the corresponding rows and columns are deleted from the stiffness matrix and the load vector. This leaves a stiffness matrix and a load vector having only free degrees of freedom, $\left[K_{ff}\right]$ and $\left\{F_f\right\}$ respectively. The solution of the equation is the final step in analysis:

$$\left\{F_f\right\} = \left[K_{ff}\right] \cdot \left\{D_f\right\} \Rightarrow \left\{D_f\right\} = \left[K_{ff}\right]^{-1} \cdot \left\{F_f\right\} \tag{4.15}$$

The (zero) displacements for the stationary degrees of freedom are added back to the result, thus creating the displacement vector {D}.

4.2 DISPLACEMENT, STRAIN AND STRESS FIELD

4.2.1 Displacement

After the solution of the equation, control point displacements are obtained. Unlike classical FEM, control points are usually placed outside the area of the model. In general, displacements of the model differ from the displacements of the corresponding control points. Conclusively, analysis results are considered "pseudo-displacements" and play an auxiliary role in calculating the physical model ones. As mentioned before, the distribution of the displacement field is achieved via shape functions:

1D:

$$\mathrm{d}(\xi)=\sum_{i=1}^{N}\left\{\mathrm{N_i}\left(\xi\right)\cdot\mathrm{D_i}\right\}=\underset{(1\times N)}{\left\{\mathrm{N_i}\left(\xi\right)\right\}^{\mathrm{T}}}\cdot\underset{(N\times 1)}{\left\{\mathrm{D}\right\}} \tag{4.16}$$

2D:

$$\mathrm{d}(\xi,\eta)=\sum_{i=1}^{N}\left\{\mathrm{N_i}\left(\xi,\eta\right)\cdot\mathrm{D_i}\right\}=\underset{(1\times N)}{\left\{\mathrm{N_i}\left(\xi,\eta\right)\right\}^{\mathrm{T}}}\cdot\underset{(N\times 1)}{\left\{\mathrm{D}\right\}} \tag{4.17}$$

3D:

$$\mathrm{d}(\xi,\eta,\zeta)=\sum_{i=1}^{N}\left\{\mathrm{N_i}\left(\xi,\eta,\zeta\right)\cdot\mathrm{D_i}\right\}=\underset{(1\times N)}{\left\{\mathrm{N_i}\left(\xi,\eta,\zeta\right)\right\}^{\mathrm{T}}}\cdot\underset{(N\times 1)}{\left\{\mathrm{D}\right\}} \tag{4.18}$$

where:

- $\mathrm{N_i}$ is the shape function i
- $\mathrm{D_i}$ is the displacement of the corresponding control point
- N is the total number of control points

If a control point c is interpolatory to the curve at $\left(\xi_c,\eta_c,\zeta_c\right)$, it follows that:

$$\mathrm{d}(\xi_c,\eta_c,\zeta_c)=\sum_{i=1}^{N}\left\{\mathrm{R_i}\left(\xi_c,\eta_c,\zeta_c\right)\cdot\mathrm{D_i}\right\}=1\cdot\mathrm{D_c}=\mathrm{D_c} \tag{4.19}$$

.

Displacements of interpolatory control points are physical model's displacements as well.

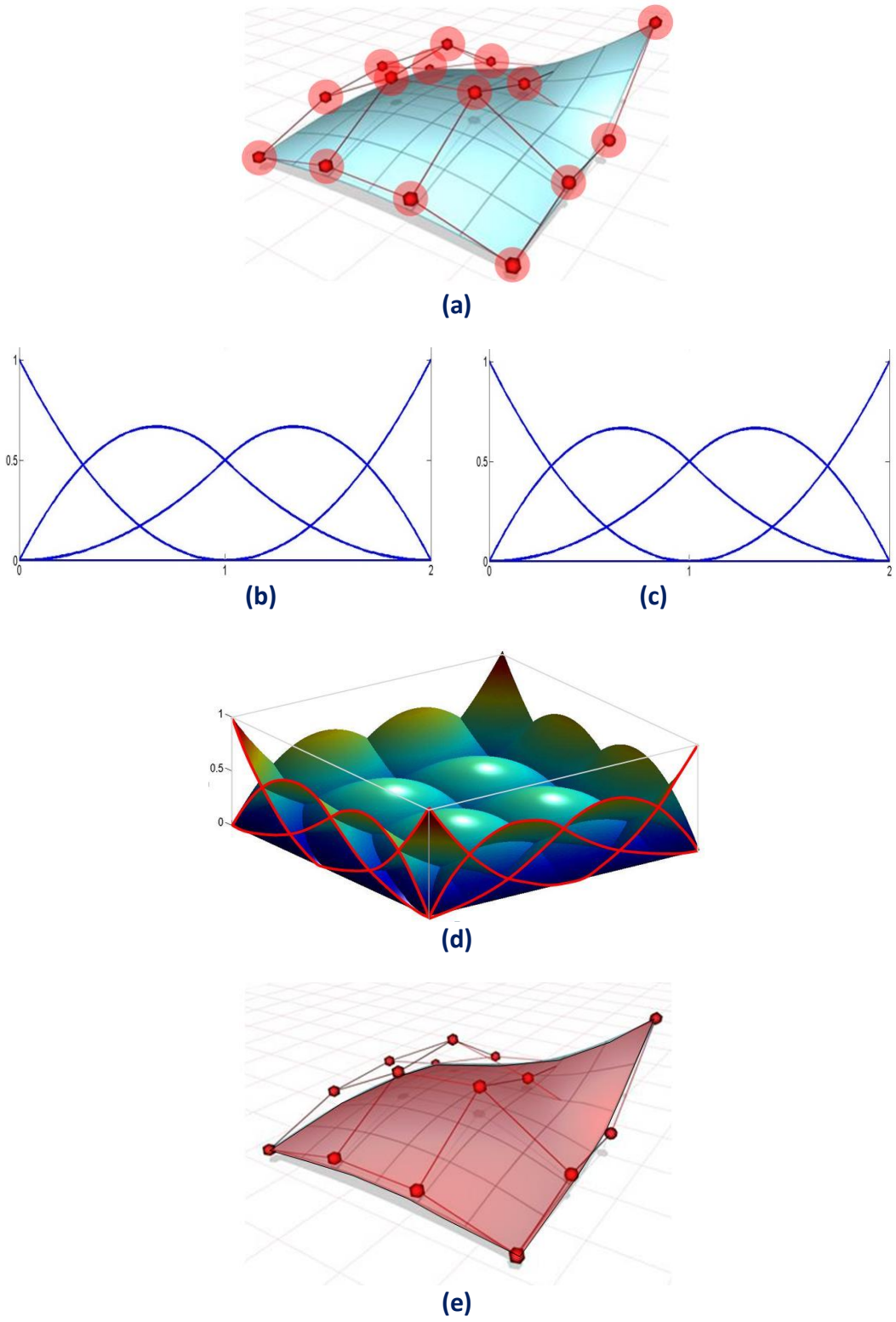


Figure 4.1.
(a) Displacement solution is found on control points. **(b),(c)** Basis functions of axis ξ and η respectively**.(d)** Corresponding 2D shape functions derived from full tensor product of ξ,η basis functions. **.(e)** Displacement field calculated combining nodal displacements and shape functions
(Iakovos Antonios 2015).

4.2.2 Stress and Strain

The strain vector can be evaluated at any point in the field with the help of control point displacements and the deformation matrix [B]:

1D

$$\underset{(1\times1)}{\{\varepsilon(\xi)\}} = \underset{(1\times N)}{[B(\xi)]} \cdot \underset{(N\times1)}{\{D\}} \tag{4.20}$$

2D

$$\underset{(3\times1)}{\{\varepsilon(\xi,\eta)\}} = \underset{(3\times2N)}{[B(\xi,\eta)]} \cdot \underset{(2N\times1)}{\{D\}} \tag{4.21}$$

3D

$$\underset{(6\times1)}{\{\varepsilon(\xi,\eta,\zeta)\}} = \underset{(6\times3N)}{[B(\xi,\eta,\zeta)]} \cdot \underset{(3N\times1)}{\{D\}} \tag{4.22}$$

Applying Hooke's constitutive law leads to:

1D

$$\underset{(1\times1)}{\{\sigma(\xi)\}} = \underset{(1\times1)}{[E]} \cdot \underset{(1\times1)}{\{\varepsilon(\xi)\}} = \underset{(1\times1)}{[E]} \cdot \underset{(1\times N)}{[B(\xi)]} \cdot \underset{(N\times1)}{\{D\}} \tag{4.23}$$

2D

$$\underset{(3\times1)}{\{\sigma(\xi,\eta)\}} = \underset{(3\times3)}{[E]} \cdot \underset{(3\times1)}{\{\varepsilon(\xi,\eta)\}} = \underset{(3\times3)}{[E]} \cdot \underset{(3\times2N)}{[B(\xi,\eta)]} \cdot \underset{(2N\times1)}{\{D\}} \tag{4.24}$$

3D

$$\underset{(6\times1)}{\{\sigma(\xi,\eta,\zeta)\}} = \underset{(6\times6)}{[E]} \cdot \underset{(6\times1)}{\{\varepsilon(\xi,\eta,\zeta)\}} = \underset{(6\times6)}{[E]} \cdot \underset{(6\times3N)}{[B(\xi,\eta,\zeta)]} \cdot \underset{(3N\times1)}{\{D\}} \tag{4.25}$$

Note that stress and strain vectors are evaluated via the derivatives of the shape functions. This means that their distribution is going to be one order less than the displacement distribution. This is why stress and strain continuity cannot be achieved in FEM models, where shape functions are always C^{-1} continuous. This problem is solved when the derivatives of the shape functions are also continuous, which means using shape functions with C^{1} continuity or higher.

5. UNIFORM REFINEMENT

5.1 INTRODUCTION

NURBS geometries lie at the forefront of designing technology as they provide numerous possibilities for representation, while supplying a reliable basis which can be easily adjusted for the purpose of analysis. Thus, it is commonplace to use a more complex mesh to obtain better analysis results. The transition from a coarse mesh to a fine mesh preserves the geometrical features of the model, while providing the designer the ability to modify specific parts of the entity. In order to efficiently use refinement in analysis, the engineer has to understand the principles involved and the ways in which the basis is altered. Refinement is an automated but complicated process that requires minimal manual effort.

The combination of a limited number of control points with a small polynomial order is the optimum solution for a designer. This coarse mesh provides exact geometrical representation with reduced computational cost. It provides a certain level of understanding and control for the user, who can comprehend the simplified patterns of the basis and control net. The creation of stiffness matrix is also less time-consuming, due to the limited number of degrees of freedom and interconnections involved.

The coarse mesh, however, has flexibility issues. This means that each control point affects a rather large part of the model, so that small changes to control point variables reflect to significant changes in geometry. Moreover, basis function overlapping is reduced in coarse meshes. This makes a coarse mesh unsuitable for analysis. A feasible approach is the introduction of a coarse mesh for design, and afterwards refinement of this mesh for analysis purposes.

With the creation of a finer mesh, a detailed, flexible control net is introduced. Each control point affects a smaller portion of the model and the number of interconnections is usually increased. Mapping from parameter to physical space remains unchanged. This is very important in terms of analysis. Stiffness matrix calculation is more time-consuming, but accuracy per degree of freedom is also improved.

There are three ways in which a B-Spline basis can be enriched. A different knot value vector may be selected, the polynomial order may be increased or a combination of both may occur when the raise of the polynomial order is followed by the introduction of a richer knot vector. These techniques are referred to as h-, p- and k- refinement respectively.

5.2 KNOT VALUE INSERTION

Knot value insertion is the introduction of a finer mesh by enrichment of the existing knot value vector Ξ. A new knot value vector Ξ* is created, such that $\Xi \subset \Xi^*$. Only internal knot values can be added; the boundaries have to remain intact. A new set of B-Spline basis functions is created. The coarse mesh representation is evaluated, as usual, by:

$$C(\xi) = \sum_{j=1}^{n} \left\{ N_{j,p}(\xi) \cdot X_j \right\} = \underset{1\times n}{\left\{ N(\xi) \right\}^T} \cdot \underset{n\times 1}{\left\{ X \right\}} \tag{5.1}$$

whereas the new representation

$$C(\xi) = \sum_{i=1}^{m} \left\{ N^*_{i,p}(\xi) \cdot X^*_i \right\} = \underset{(1\times m)}{\left\{ N^*(\xi) \right\}^T} \cdot \underset{(m\times 1)}{\left\{ X^* \right\}} \tag{5.2}$$

Therefore, the new set of control points $\left\{ \overline{X} \right\}$ has to be defined. This can be achieved with the evaluation of a transformation matrix, so that:

$$\underset{(mx1)}{\left\{ X^* \right\}} = \underset{(m\times n)}{\left[T^p \right]} \cdot \underset{(n\times 1)}{\left\{ X \right\}} \tag{5.3}$$

This matrix is formed recursively:

$$T^0_{ij} = \begin{cases} 1, & \bar{\xi}_i \in \left[\xi_{j,} \xi_{j+1} \right) \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

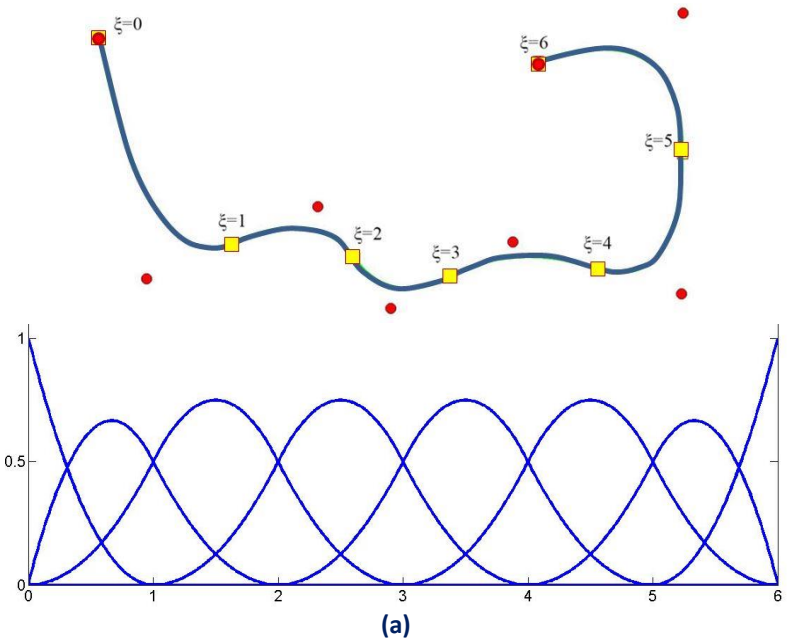
$$T^q_{i,j} = \frac{\xi^*_{i+q} - \xi_j}{\xi_{j+q} - \xi} \cdot T^{q-1}_{i,j} + \frac{\xi_{j+q+1} - \xi^*_{i+q}}{\xi_{j+q+1} - \xi_{j+1}} \cdot T^{q-1}_{i,j+1} \tag{5.5}$$

$$\text{, for } q = 1, 2, \ldots, p$$

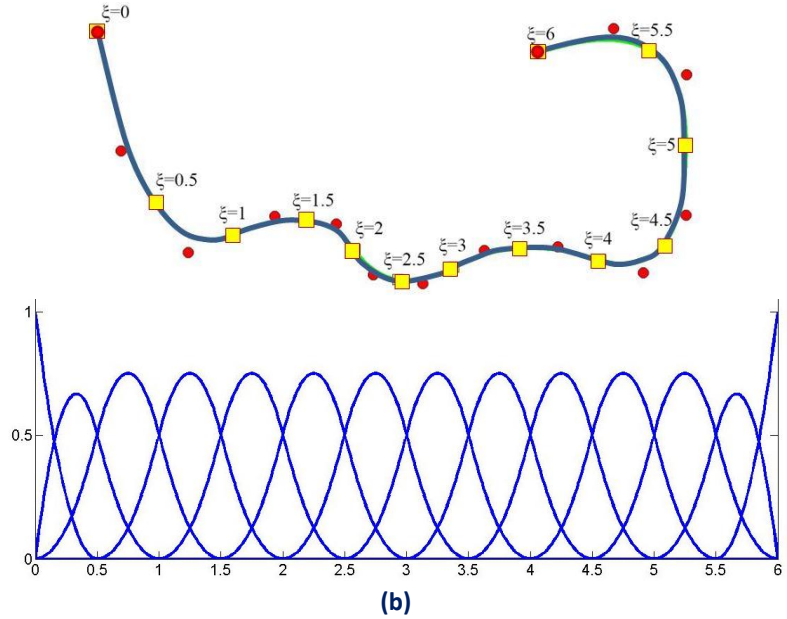
This technique is called h-refinement.

The B-Spline curve represented in **Fig. 5.1.a** is refined by knot value insertion in **Fig. 5.1.b**. Both geometry and parametric mapping remain intact. This can be confirmed by the fact that already existing knots have not been moved. For each new knot value, a basis function and a corresponding control point have been created. The support is still p+1 knot value spans, but their size has been reduced by the insertion of new knot values. Therefore, each control point now affects a much

smaller part of the curve. The limited area of effect for each control point leads to a more accurate approximation to the curve.

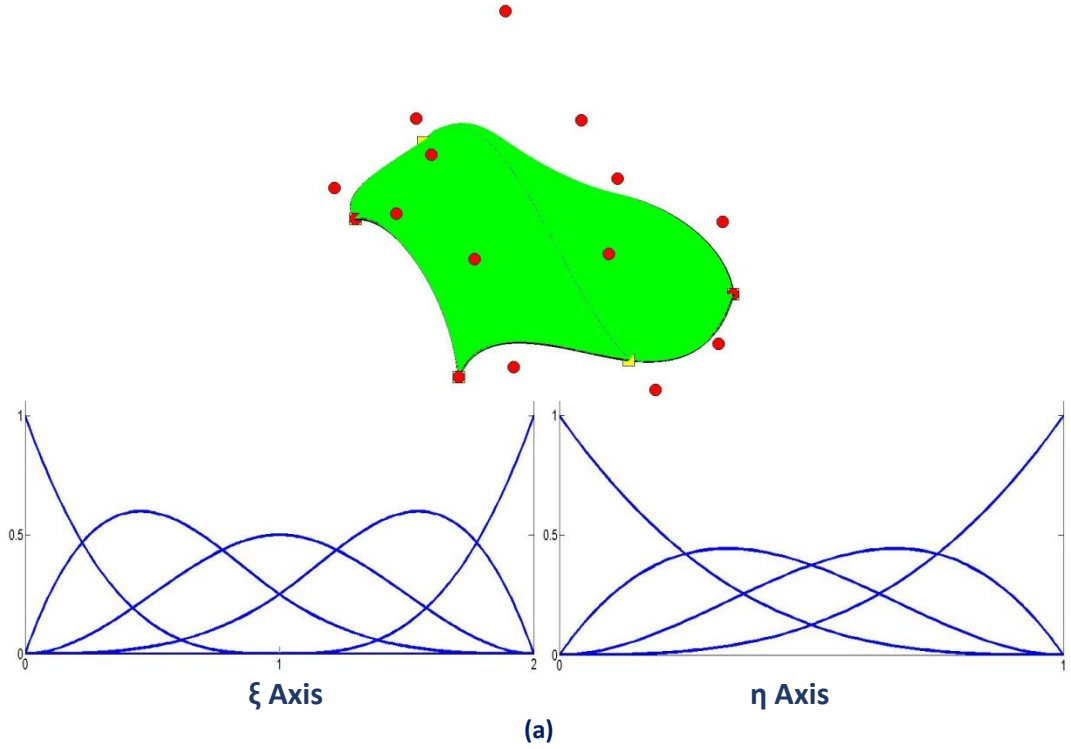


(a)



(b)

Figure 5.1.
h-refinement applied on a B-Spline curve.**(a)** Coarse Mesh. Knot Value Vector: $\Xi = \{0 \;\; 0 \;\; 0 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 6 \;\; 6\}$
(b) Fine Mesh. Knot Value Vector: $\bar{\Xi} = \{0 \;\; 0 \;\; 0 \;\; 0.5 \;\; 1 \;\; 1.5 \;\; 2 \;\; 2.5 \;\; 3 \;\; 3.5 \;\; 4 \;\; 4.5 \;\; 5 \;\; 5.5 \;\; 6 \;\; 6 \;\; 6\}$
(Iakovos Antonios 2015)



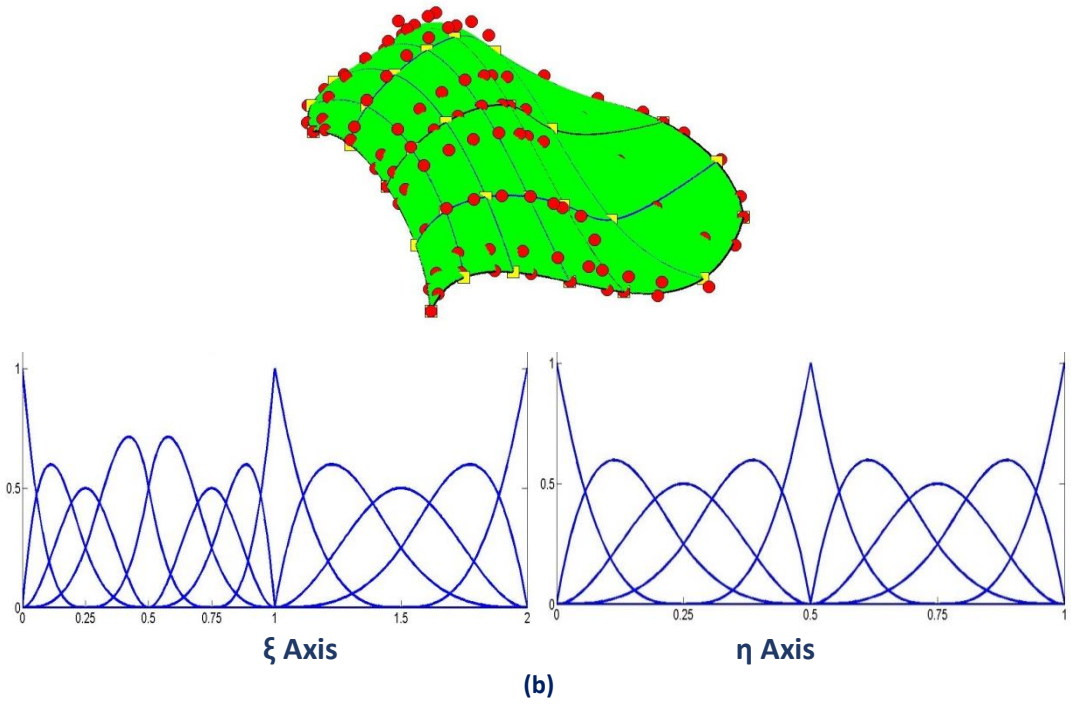


Figure 5.2.
Knot Insertion in multiple directions.**(a)** Coarse Mesh. Knot Value Vectors:

$$\Xi = \{0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2\}$$

$$H = \{0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1\}$$

(b) Fine Mesh. Knot Value Vectors:

$$\Xi = \{0 \quad 0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.5 \quad 0.75 \quad 1 \quad 1 \quad 1 \quad 1.5 \quad 2 \quad 2 \quad 2 \quad 2\}$$

$$H = \{0 \quad 0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.5 \quad 0.5 \quad 0.75 \quad 1 \quad 1 \quad 1 \quad 1\}$$

(Iakovos Antonios 2015)

Refinement is applicable in multi-directional problems as well. The surface in **Fig. 5.2**. is refined both per ξ and η axes. **Fig. 5.2** represents basis functions before and after refinement.

Note that for every univariate (1D) control point per η, a whole set of control points per ξ are defined. Each set is refined individually, as shown in **Fig. 5.3.a** and **Fig. 5.3.b**. After the insertion of new control points per ξ, the same process is followed for refinement per η. The order in which refinement is applied on the parametric axes is of no importance; due to tensor product properties.

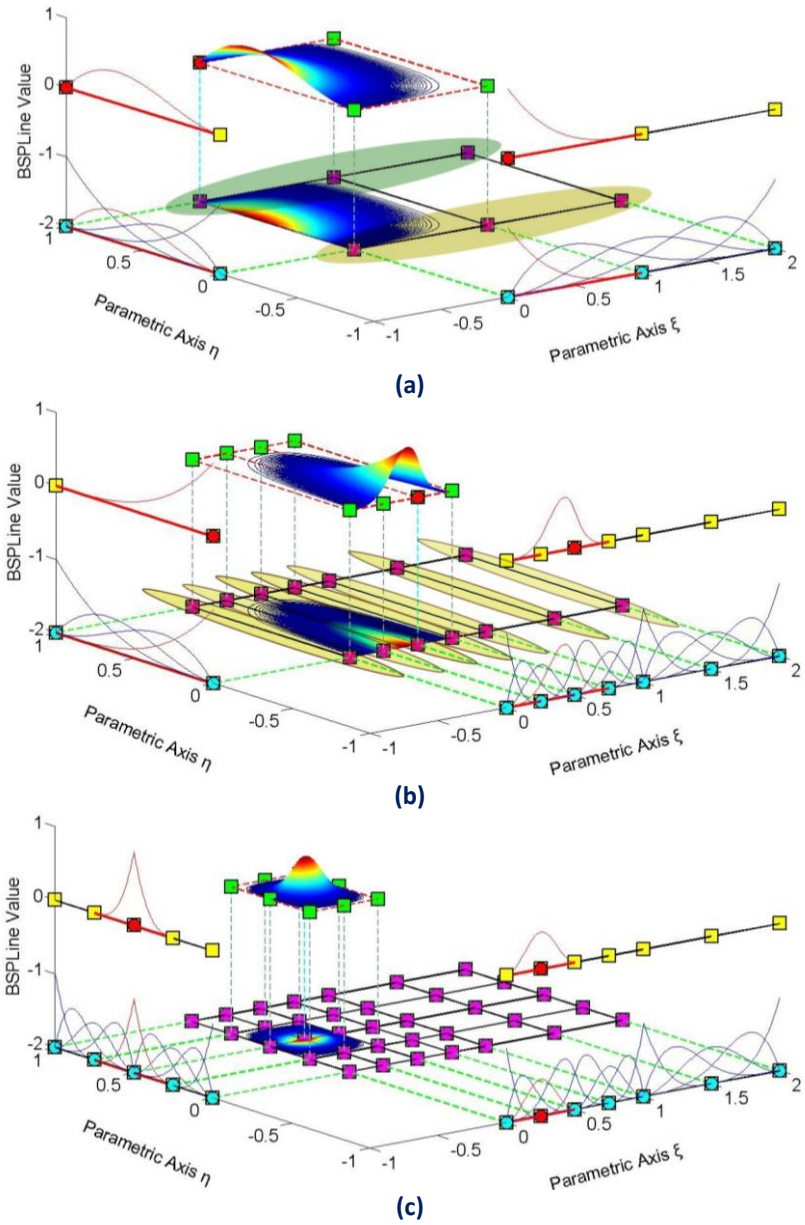


Figure 5.3.
Basis and Shape Functions for Surface Coarse and Fine Mesh **(a)** Coarse Mesh. **(b)** Refinement per ξ direction.**(c)** Refinement per η direction. Fine mesh.

5.3 DEGREE ELEVATION

Instead of adding knot values to an existing knot value vector, the increment of the polynomial order can also enrich the basis. Apart from geometry, the mapping from parameter to physical space must also remain unchanged. This is achieved by keeping the same continuity per knot for both coarse and fine mesh.

In order to increase the degree of a B-Spline curve from p to p*, the new knot value vector has to be defined first. No new knots are added, but every existing knot's multiplicity is increased by p*-p times. The coarse mesh representation is defined as:

$$C(\xi) = \sum_{j=1}^{n} \left\{ N_{j,p}(\xi) \cdot X_j \right\} = \underset{(1\times n)}{\left\{ N(\xi) \right\}^T} \cdot \underset{(n\times 1)}{\left\{ X \right\}} \tag{5.6}$$

whereas the fine mesh representation as:

$$C(\xi) = \sum_{i=1}^{m} \left\{ N^*_{i,p}(\xi) \cdot X^*_i \right\} = \underset{(1\times m)}{\left\{ N^*(\xi) \right\}^T} \cdot \underset{(m\times 1)}{\left\{ X^* \right\}} \tag{5.7}$$

This leads to:

$$\underset{(1\times m)}{\left\{ N^*(\xi) \right\}^T} \cdot \underset{(m\times 1)}{\left\{ X^* \right\}} = \underset{(1\times n)}{\left\{ N(\xi) \right\}^T} \cdot \underset{(n\times 1)}{\left\{ X \right\}} \tag{5.8}$$

The new set of control points {X*}, necessary for the representation, will be defined through a transformation matrix for p-refinement. There are many efficient algorithms for degree elevation. The following is a quick, reliable approach that can be efficiently used.

At first, coarse mesh B-Spline basis functions are evaluated at m points throughout the patch. Careful selection (such as no points of C^0 continuity or other irregularities are involved) is preferred. This way, a B-Spline function matrix [N] is created which contains the values of the n basis functions for the m selected points.

The same points are used for the evaluation of new basis functions, with the refined degree and the new knot value vector. Thus, [N*] is created. Since the same points are evaluated and the parametric mapping remains the same, it applies that:

$$\underset{(m\times m)}{\left[N^* \right]^T} \cdot \underset{(n\times 1)}{\left\{ X^* \right\}} = \underset{(m\times n)}{\left[N \right]^T} \cdot \underset{(n\times 1)}{\left\{ X \right\}} \Rightarrow \underset{(n\times 1)}{\left\{ X^* \right\}} = \left(\underset{(m\times m)}{\left[N^* \right]^T} \right)^{-1} \cdot \underset{(m\times n)}{\left[N \right]^T} \cdot \underset{(n\times 1)}{\left\{ X \right\}} \tag{5.9}$$

Therefore,

$$\underset{(m\times n)}{[T]} = \left(\underset{(m\times m)}{\left[\overline{N}\right]^T} \right)^{-1} \cdot \underset{(m\times n)}{[N]^T} \tag{5.10}$$

If $[N^*]^T$ cannot be reversed, a different set of points have to be selected. This procedure can also be used for h-refinement applications.

The curve in **Fig. 5.4** is subjected to p-refinement. Note that knot spans are unchanged. Geometry and mapping from parameter space remain intact in p-refinement as well. Continuity remains C^1 across every internal knot. Curve approximation by control points is also improved with p-refinement.

Tensor product properties prove that p-refinement can be utilized in multiple directions.

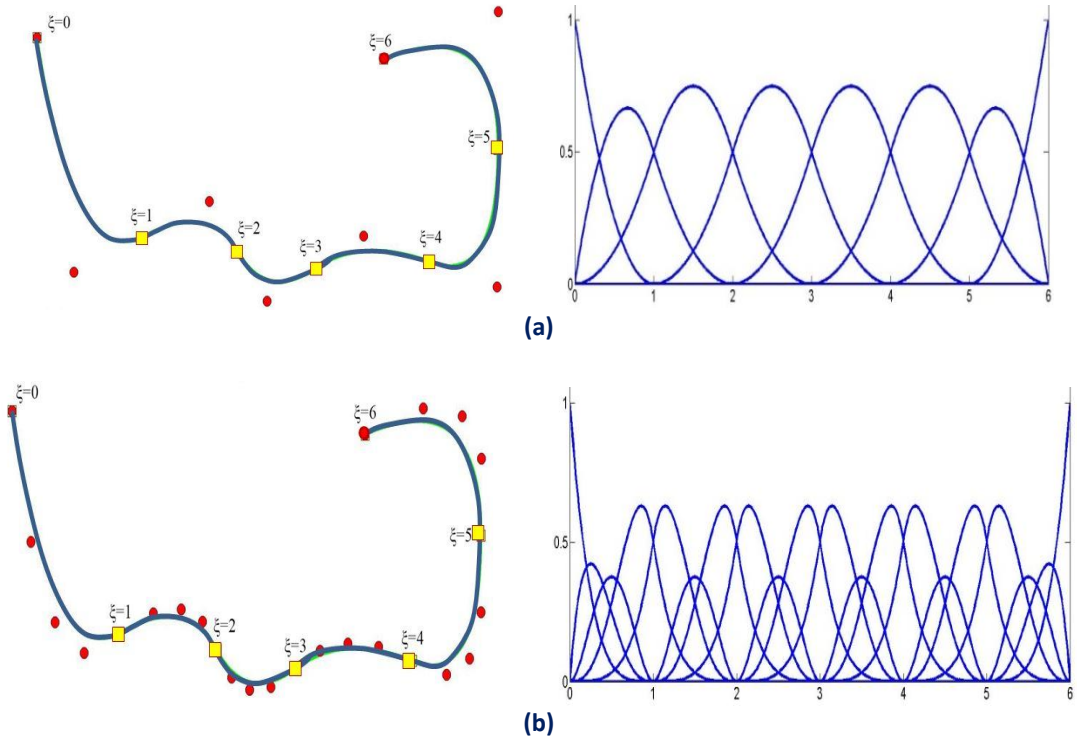


Figure 5.4.
p-refinement on a B-Spline curve, from p=2 to p=4.**(a)** Coarse mesh. knot value vector: $\Xi = \{0\ 0\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 6\ 6\}$ **(b)** Fine mesh. knot value vector:
$\Xi = \{0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 2\ 2\ 2\ .\ .\ .\ 6\ 6\ 6\ 6\ 6\}$

5.4 Degree Elevation and Knot Insertion

Increasing polynomial degree by p-refinement is an improvement to the basis, but continuity remains the same as in the coarse mesh. In order to improve this aspect, k-refinement was introduced by Hughes. The basic idea is that, after p-refinement, h-refinement can be applied in order to create basis functions of C^{p-1} continuity. This is a powerful tool that can lead to greater convergence rates for our models.

In **Fig. 5.5**, p-refinement is performed first, so the order is elevated to p*=3. The C^1 continuity that existed in the coarse mesh is maintained. Afterwards, h-refinement is applied, with the insertion of two extra knot values. Continuity across this knot values is C^2, taking greater advantage of the new polynomial order. The combination of p- and h- refinement brings their best features together. This is why k-refinement is considered so effective.

After p-refinement, internal knots still possess C^1 continuity. Note that only two basis functions are non-zero across every knot. After h-refinement, two new C^2 continuity knots are introduced. Three B-Spline basis functions are non-zero across these knots, ξ=2.5 and ξ=3.5. This way, greater levels of continuity for the new polynomial order are utilized.

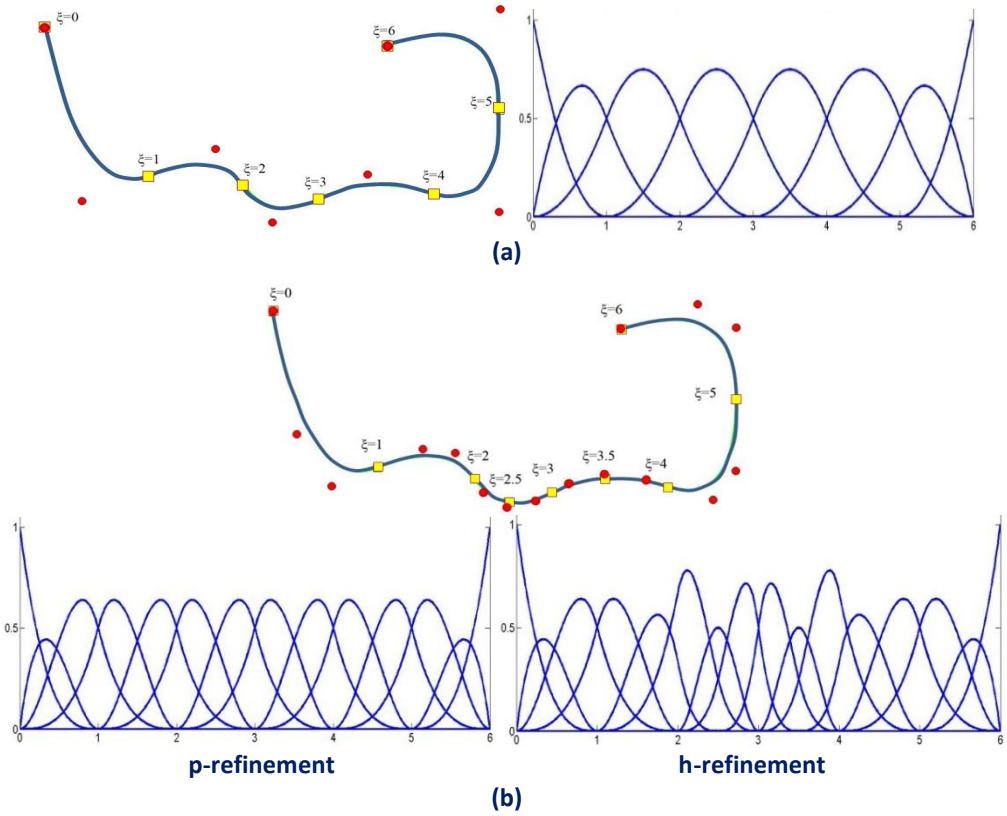


Figure 5.5.
k-refinement on a B-Spline curve. **(a)** Coarse mesh. Ξ={0 0 0 1 2 3 4 5 6 6 6}
(b) Fine mesh. Polynomial order p=3. Knot value vector B-Spline basis functions for the two stages of k-Refinement. p-refinement to $p=3$. Knot value vector: Ξ ={0 0 0 0 1 1 2 2 3 3 4 4 5 5 6 6 6 6}
h-refinement. knot value vector: Ξ = {0 0 0 0 1 1 2 2 2.5 3 3 3.5 4 4 5 5 6 6 6 6}

5.5 REFINEMENT WITH NURBS

Everything mentioned about refinement is applicable to B-Spline entities. The question is what happens with NURBS. Different weight values and the complexity of NURBS shape functions prevent the refinement straight at the NURBS entity. However, every NURBS is created from the projection of a B-Spline; therefore, NURBS refinement can be achieved by refining the corresponding projective B-Spline curve.

The first step is to evaluate the projective control points $\{B^w\}$, by multiplying every coordinate with the corresponding weight, as shown below:

$$\left\{X_i^w \quad Y_i^w \quad Z_i^w\right\} = \left\{X_i \cdot W_i \quad Y_i \cdot W_i \quad Z_i \cdot W_i\right\} \tag{5.11}$$

Weights are the fourth-coordinate of the projective curve.

Refinement is applied for each set of the four coordinates, $\{X^w\ Y^w\ Z^w\ W\}$. Afterwards, updated NURBS control point coordinates and weights are obtained by dividing the Cartesian coordinates with the weights:

$$\left\{X_i \quad Y_i \quad Z_i\right\} = \left\{\frac{X_i^w}{W_i} \quad \frac{Y_i^w}{W_i} \quad \frac{Z_i^w}{W_i}\right\} \tag{5.12}$$

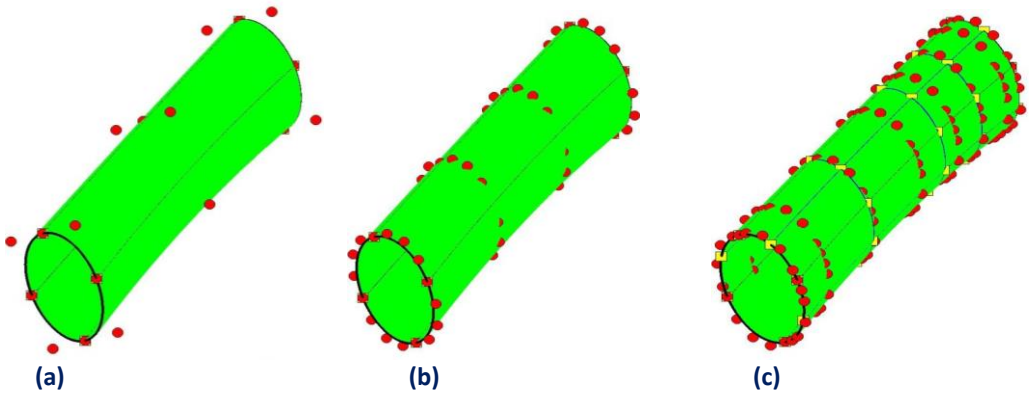


Figure 5.6.
(a) Coarse mesh and **(b)** p-refinement as a part of k-refinement **(c)** k-refinement for a NURBS surface.

5.6 DRAWBACKS OF GLOBAL REFINEMENT

In contrast with classical FEA implementations, in isogeometric analysis with B-Splines and NURBS the refinement procedure doesn't always lead to better and more accurate results. Especially in the h-refinement technique this phenomenon is of even greater importance. A serious matter occurs when the tensor product B-

Splines and their corresponding NURBS are refined. Depending on the geometry of each structure that is analyzed, refinement for each of the parameter space axes may cause serious analysis instability. Due to the full tensor product nature the produced elements and the corresponding shape functions may present an unsuitable dimension ratio that may be up to 1:10. As we know from the classical FEA and the isoparametric concept that is being also used in IGA, Jacobian matrix is unable to represent properly the geometry in the physical space when such elements and dimensions occur in parameter space.

Even if h-refinement which refers to knot insertion is not the case, p- and k-refinement have also drawbacks. In most problems the error analysis will indicate small regions of high error that need to be recomputed and re-meshed. In large scale problems with thousands or millions degrees of freedom the classical refinement techniques will eventually lead in an enormous increase of the degrees of freedom. This increase is for most cases unefficient in terms of the estimated CPU time, as it introduces new sets of control points and knots in and out of the refined domain.

6. HIERARCHICAL REFINEMENT

6.1 INTRODUCTION

The fact that classical FEA implementations and by extension IGA are both numerical methods that provide approximate solutions leads always to solution errors. These errors may appear in different areas of the structure such as edges, interfaces, discontinuities, cracks and in regions of significantly altering geometry.

In order to estimate these errors, many error-estimation techniques have been proposed. These techniques are mainly based on energy theorems and norms to calculate the error of each element or even each basis function. Most of these error estimators are a-posteriori, which means that the error is calculated after one at least analysis iteration. Researchers worldwide are trying to propose a sufficient and stable error-estimator appropriate for IGA implementations. However, up to the day this thesis is being written, there is no clear answer which error-estimation technique is the most suitable for adaptive algorithms and hierarchical refinement implementations in general.

After estimating the errors and locating the areas they appear, there is need of some techniques to improve the solution. Three refinement techniques where explained in the previous chapter. In the lines following a brief review of these three methods is presented.

- **h-refinement**
  This is the most common refinement type that refers to knot insertion in the basis of each axis in the parameter space. The major drawback of this technique is that due to the tensor-product nature of B-Splines, unsuitable analysis meshes with bad element dimensions ratio may occur in 2D or 3D problems.

- **p-refinement**
  It refers to degree elevation of the basis in order to increase the support and interaction of the basis functions and thus achieve better results. Overlapping and bandwidth of stiffness matrix are also increasing.

- **k-refinement**
  Last but not least follows the combination of the two previous methods, k-refinement, which is a mechanism that exists only in isogeometric analysis. K-refinement tries to combine successfully p and h refinement in order to achieve faster and more precise results. Usually p-refinement is applied before h-refinement in implementations in order to achieve better results.

As it was already explained the main drawback of these techniques is their full tensor product nature. A more recent and interesting technique was the hierarchical refinement proposed by Vuong et al based on the Phd thesis of Kraft (1997). Kraft developed a methodology for computational geometry applications in order to achieve local refinement of geometry meshes. What was innovating about this method, was that refinement could be obtained locally, overcoming completely the obstacle of the tensor product nature. Local refinement in isogeometric analysis was until then a feature strictly connected to T-Splines. Briefly, the main idea of Vuong et al was to directly replace the coarse basis functions in the region of significant errors with a new set of finer basis-functions as Kraft introduced. This simple idea can be applied by respecting the restriction of a nested nature between the levels of refinement as Kraft first proposed.

The local nature of this kind of refinement as it is shown in **Fig. 6.1** allows the reduction of errors by inserting fewer new degrees of freedom than those corresponding to the full tensor-product refinement. This fact suggests that CPU time can be preserved. But it is not as simple as it seems, as what really matters is the total number of degrees of freedom in relation to the precision achieved, a crucial parameter that needs to be examined thoroughly. Using this as the basis for further research, Giannelli, Schillinger, Bornemann-Cirak, Wu-Huang-Liu-Zuan, Johannessen-Kvamsdal have all proposed their own variations and methodologies. In general, hierarchical refinement is currently a state of the art subject of research and much effort has been made by the engineering society in order to use it in more efficient and systematic ways.

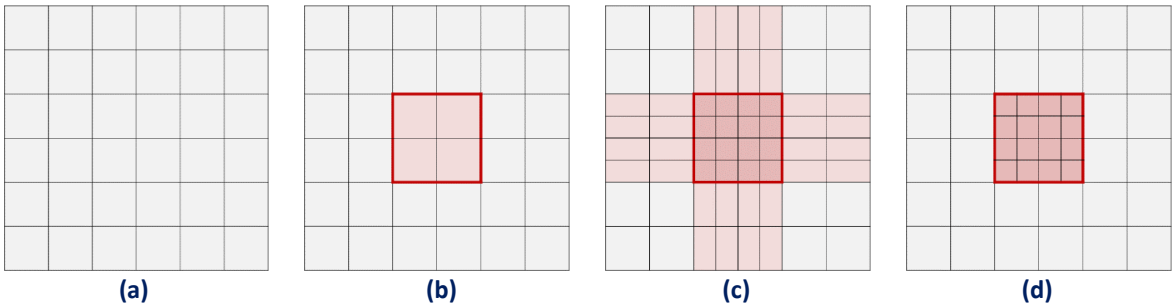


Figure 6.1.
For a tensor product 2D space in **(a)**, we are interested in refining the area highlighted in **(b).** In **(c)** it is obvious that we cannot directly refine this region, when applying tensor product h-refinement, as it refines regions that we don't want to and thus, creating elements with bad size ratio. In **(d)** the selected region is hierarchically locally refined and thus the mesh is denser for the region of interest.
(Iakovos Antonios, 2015)

To explain and understand Vuong's proposed refinement technique the theoretical foundations of this particular method like the Kraft algorithm and the subdivision property need to be introduced. After pointing out the main features that need to be comprehended by the reader, other proposed methods of hierarchical refinement will be explained, constructed and evaluated by their advantages and disadvantages in the following chapter. In order to support the theoretical base of each technique some lemmas and proofs will be presented. Key properties of B-Splines turn out to be essential for the successful use of hierarchical refinement, such as locality, linear independence, partition of unity, nested spaces, representability of geometry and unified representation.

6.2 BASIC INGREDIENTS

6.2.1 Kraft Algorithm

6.2.1.1 Tensor Product B-Spline Spaces

Firstly, the properties and the definitions that accompany tensor-product B-Splines should be clarified. The properties of a bivariate tensor-product B-Spline will be presented in the following pages. These properties can of course be generalized for the 3D case.

A bivariate tensor-product B-Spline space ***B*** is defined by specifying the polynomial degree (p, q) and the horizontal and vertical knots vectors:

$$\Xi = \left\{ \xi_0 \le \xi_1 \le \ldots \le \xi_{n+p+1} \right\} \tag{6.1}$$

$$H = \left\{ \eta_0 \le \eta_1 \le \ldots \le \eta_{m+q+1} \right\} \tag{6.2}$$

Which contain non-decreasing parametric real values so that,

$$0 \le \mu(\Xi, \xi) \le p+1 \tag{6.3}$$

$$0 \le \mu(\mathrm{H}, \eta) \le q+1 \tag{6.4}$$

are the multiplicities of the parameter values in the knot vectors (the multiplicity μ(X, x) is zero if the given value x is not a knot in X).

The space ***B*** is spanned by the tensor-product B-Splines.

$$N_{i,j} = N_{i,p,\Xi}(\xi) N_{\mathrm{j},q,\mathrm{H}}(\eta) \tag{6.5}$$

In order to represent basis functions in the parameter space we can consider different types of anchors like those presented in **Fig. 6.2**, namely:

I. naive anchors:

- if both degrees are even, each basis function is identified with the central elementary cell of its support,
- if both degrees are odd, each basis function is identified with the central point (intersection of knot lines) of its support;

- in case of mixed degrees (even/odd), each basis function is identified with the central edge of its support;

II. Greville abscissae: the (ξ, η) coordinates associated to each basis function $N_{i,j}$ are defined as averages of subsequent knots.

$$u_i = \frac{1}{p}\sum_{k=1}^{p}\xi_{i+k} \tag{6.6}$$

and

$$v_j = \frac{1}{q}\sum_{k=1}^{q}\eta_{j+k} \tag{6.7}$$

Which they satisfy the following relations:

$$\xi = \sum_{i=0}^{n} u_i N_{i,p,\Xi}(\xi) \tag{6.8}$$

$$\eta = \sum_{j=0}^{m} v_j N_{\mathrm{j,q,H}}(\eta) \tag{6.9}$$

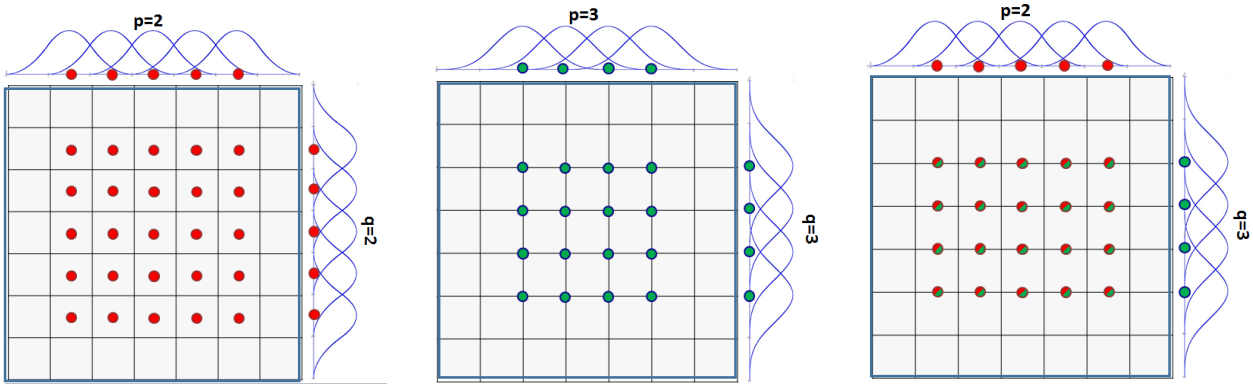


Figure 6.2.

Greville abscissae for three cases. p=q=2, p=2 q=3 and p=3 q=3
(Iakovos Antonios and Karras Dimitrios, 2015)

6.2.1.2 The Kraft Algorithm Proposal

Kraft presents in his work “Adaptive and Linear Independent Multi-Level B-Splines” (1997) a new methodology for successfully handling multi-level local refinement of B-Splines conserving at the same time the linear independency of their basis.

The basic new feature of this approach is a simple selection mechanism for B-Splines, which ensures their linear independence, while allowing complete local control of the refinement procedure. The use of B-Splines in the isogeometric approach leads to an explicit correlation between the mesh of B-Spline knots and the surface represented. Moreover, the multilevel B-Splines are similarly stable as normal tensor product B-Splines. The multilevel and hierarchical B-Spline basis is weakly stable. The resulting smooth Spline surface has the same approximation properties as discontinuous polynomial approximations. Furthermore, with an appropriate multilevel and hierarchical quasi-interpolant, we get the optimal local approximation order.

It can be shown that this multilevel B-Spline space is very well suited for scattered data approximation or interpolation. The number of degrees of freedom of the hierarchical B-Spline space can be locally increased where necessary to better approximate the given data as shown in **Fig. 6.3**.

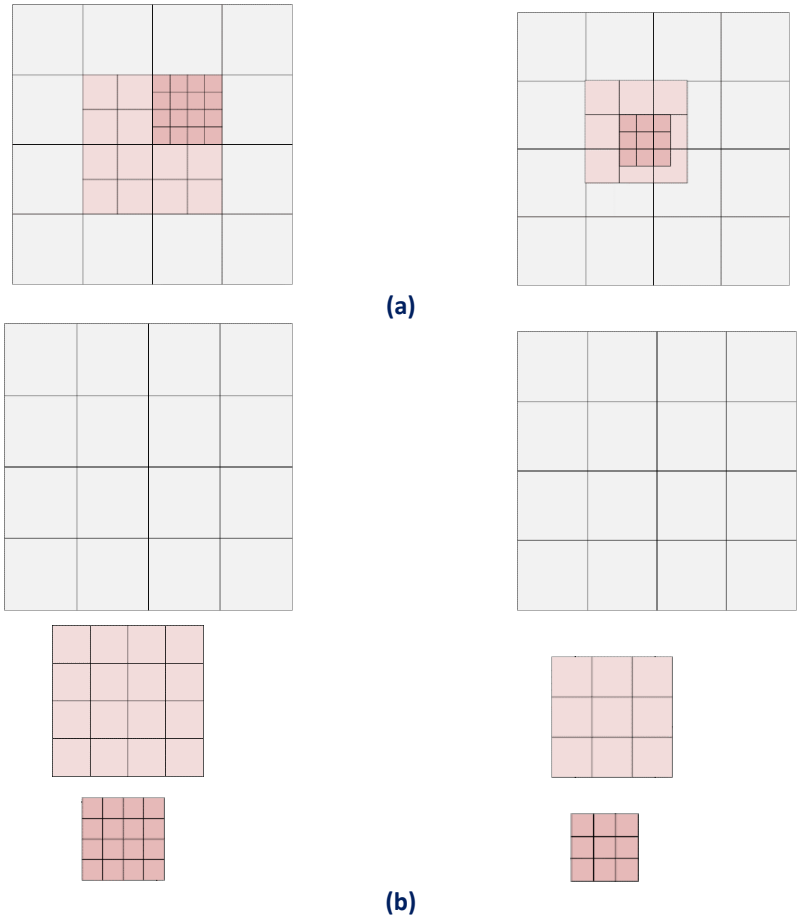


Figure 6.3.
(a) Hierarchically refined mesh for p=3 and p=2 respectively **(b)** Breakdown of the functions that consist the hierarchical meshes in (a). The support cells for three refinement levels can be seen. As the cells tend to be smaller, the support is also decreasing.
(Iakovos Antonios 2015)

6.2.1.3 Nested Spaces

We consider a dyadic sequence of grids determined by the scaled lattices

$$\left(\frac{k_1}{2^{\ell}}, \frac{k_2}{2^{\ell}}\right),$$

where ℓ denotes a random level of the hierarchy. The $h_{x1}(x)$ describes the grid-width in the direction x_i at the point $\sigma = (\xi, \eta)$. To avoid unnecessary complications we set $h_{\xi}(\xi) = h_{\eta}(\eta) = 1$. The extension of the results is straightforward. Kraft made this simplification in order to ease explanation of the algorithm. In this thesis, h will be referred as knot span length of each axis, which can differ for the two or three parametric axes and can also take values different from one.

On the grid of the dyadic level p we denote by:

$$N_p^{\ell}(\zeta) = N_{k_1,k_2}^{\ell}(\xi, \eta) = N_{\alpha}(2^{\ell}\xi - k_1) N_{\beta}(2^{\ell}\eta - k_2) \tag{6.10}$$

the B-Splines with degree (*p,q*) and open support:

$$\operatorname{supp} N_K^{\ell} = \left(\frac{k_1}{2^{\ell}}, \frac{k_1 + p + 1}{2^{\ell}}\right) \mathrm{x} \left(\frac{k_2}{2^{\ell}}, \frac{k_2 + q + 1}{2^{\ell}}\right) \tag{6.11}$$

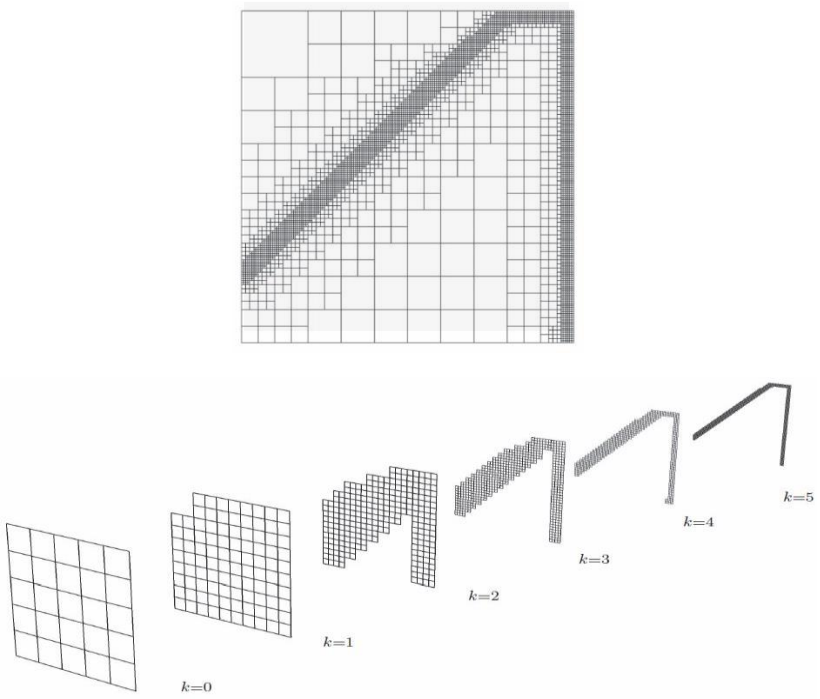


Figure 6.4.
Breakdown of a hierarchically refined mesh of totally 5 levels.
(Schillinger et al, 2012)

The recursive application of the refinement step combined with the non-intersection requirement leads to a sequence of nested refined domains as presented in the example of **Fig. 6.4**. Let Ω: = $\Omega^0 \supseteq \Omega^1 \supseteq \ldots \supseteq \Omega^n$, where n is the index of the finest level, be a nested sequence of domains with the following properties:

$$\Omega^{\ell+1} = \bigcup_{j \in J} \operatorname{supp} N_j^{\ell} : J \subseteq \mathbb{Z}^2 \tag{6.12}$$

$$\partial\Omega^{\ell} \cap \partial\Omega^{\ell+1} = 0 \tag{6.13}$$

This means that there is always one cell distance between the boundary of the coarse and fine grid

From the above properties follows that, for an appropriate set of D_{ℓ}

$$\Omega^{\ell} = \bigcup_{k \in D^{\ell}} \operatorname{supp} N_k^{\ell} \tag{6.14}$$

From the B-Splines N_k^p, k ∈ D^p we select those with support not contained in W_{p+1} as basis-functions for the hierarchical B-Spline space. In other words, we define the hierarchical B-Spline set as follows:

$$I^{\ell} := \left\{ k \in \mathbb{Z}^2 \mid \operatorname{supp} N_K^{\ell} \subseteq \Omega^{\ell}, \operatorname{supp} N_k^{\ell} \subsetneq \Omega^{\ell+1} \right\} \tag{6.15}$$

And the multilevel, hierarchical and linear B-Spline space $S_{\Omega} := S_{\Omega}^{\mathrm{p,q}} := span\left\{ B_K^{\ell} \mid \mathrm{p} \geq 0, k \in I^{\ell} \right\}$ of the degree (p, q) as the linear span of the B-Splines B_K^{ℓ} with p≥0, $k \in I^{\ell}$.

As is apparent from this definition, the B-Splines whose support lies in W_{p+1} are replaced by B-Splines B_K^p on the finer grid and thus can also be represented in the space *S*. Moreover, a spline function in *S* has the representation:

$$S : \mathbb{R}^2 \to \mathbb{R}$$

$$x \to S(\mathrm{x}) = \sum_{p>0} \sum_{k \in I^p} b_k^{\ell} N_k^{\ell}(\mathrm{x}), with : b_k^{\ell} \in \mathbb{R} \tag{6.16}$$

In order to clarify the above definitions given by Kraft, examples will be presented and discussed. Initially, a bivariate parametric space W_0 is considered with the following knot value vectors without multiplicities in their knots Ξ = H = [0 1 2 3 4 5 6 7 8 9]. The functions have degree p=q=2 (quadratic B-Splines). From the tensor product property it occurs that each B-Spline has a support that consists of (p+1)(q+1) = $(p+1)^2$, for p=q cells. For the given example the support consists of 9 cells of the parameter space mesh (grid). As it has been

mentioned before, to address efficiently the functions either naïve anchors or Greville abscissae will be used. In addition, different shapes and colors will be used in order to avoid confusion on the illustrative examples

Assuming W_1 := [2 3 4 5] × [3 4 5 6] which is the support of the B-Spline $N^0_{2,3}$ (where the top index refers to the hierarchy level and the bottom index refers to the down and left knot of the support) at the level 0. Because of (6.14), this is the smallest domain which we can use for W_1. To fulfill the condition (6.15) of the index set I^0, we have to remove the B-Spline and we have to insert 16 B-Splines at level 1. The support of the B-Spline $N^0_{2,4}$ for example, overlaps W_1 but is not a subset of it. This means that $N^0_{2,4}$ is not fully contained in it and thus it cannot be removed completely. In isogeometric analysis implementations this property isn't necessary to be valid as it will be discussed later. The explained example is depicted in **Fig. 6.5**.

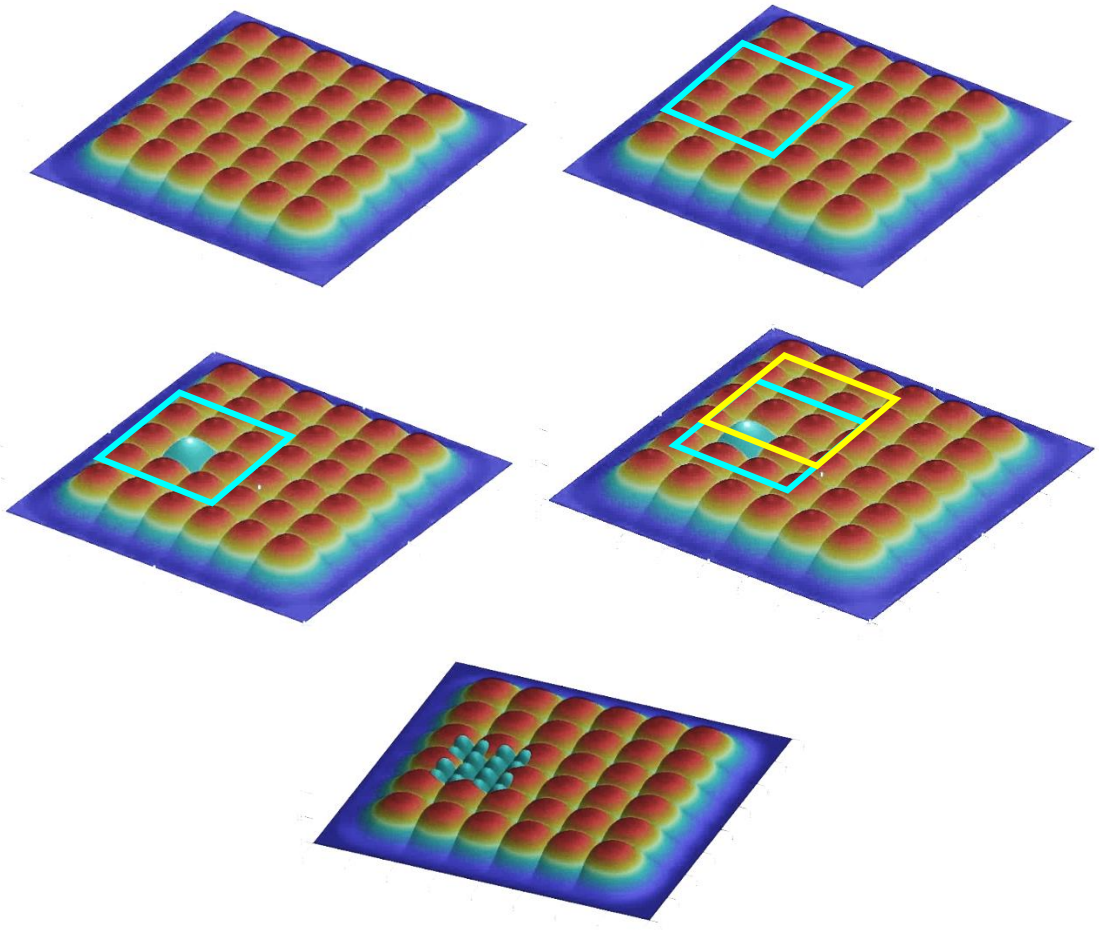


Figure 6.5.

(a) W_0 with parametric knot value vectors Ξ=H=[0 1 2 3 4 5 6 7 8 9] **(b)** W_1 [1, 2, 3, 4] x [2, 3, 4, 5]
(c) Basis function $N^0_{2,3}$ with support W_1 **(d)** Support of basis function $N^0_{2,4}$ part of W_1
(e) Removal of basis function $N^0_{2,3}$ and introduction of 16 finer basis functions
(Iakovos Antonios 2015)

6.2.1.4 Recursive Procedure

Due to the B-Spline representation, Spline approximation methods can be implemented almost as efficiently as non-smooth piecewise polynomial approximations. Optimal order of accuracy can already be obtained by computing the B-Spline coefficients as a weighted average of a small number of function values. We have constructed a hierarchical quasi interpolant which fulfills the optimal local approximation order and is based on the quasi interpolation of tensor product B-Splines.

In the next step the algorithm tests for each B-Spline in the middle area of its support if the maximum error between the given data and the spline surface is exceeding the tolerance. If so, the index of this and only this B-Spline will be removed from the hierarchical B-Spline set I_p and the indices of the B-Splines at the level $p + 1$ which are generated by subdividing are inserted into the hierarchical B-Spline set I_{p+1}. In this way the full amount of degrees of freedom of the level $p + 1$ in this area is exactly reached. So the resulting surface after the next approximation step will be exactly the same in this area as if we used a tensor-product B-Spline approximation all over W_0 with a full grid of this level. If biquadratic B-Splines are used, this local area will be the rectangle between the two grid-lines in u and v direction which lies in the support of the B-Spline. To fulfill the conditions (6.12), (6.13) and (6.15) on the domain W_p and on the index-set I_p, it must be tested, if the neighbors of these B-Splines also have to be removed. This is a local operation and therefore it is not very cost-intensive. This very important step will be later utilized in isogeometric implementations.

6.2.1.5 Conclusions

Kraft has developed a multilevel and hierarchical B-Spline space as a generalization of the normal tensor-product B-Spline space. The space consists of linearly independent B-Splines, which are defined on different grid levels and can be derived through subdivision of higher B-Splines. With this hierarchical B-Spline space, we have complete local control of the refinement of the spline space. In areas where the indices of all overlapping B-Splines of level p are members of the hierarchical B-Spline set, we have the same properties, as if we use only B-Splines of this level.

The use of B-Splines in this approach leads to an explicit correlation between the mesh of B-Spline knots and the surface represented. Because of the direct relation between the B-Spline coefficients and the spline surface, the evaluation of the B-Spline surface can be done fast and it is not cost-intensive.

The visualisation of the surface can also be done in a fast way. These properties imply a good perfomance for scattered data interpolation or approximation. Kraft has developed an iterative approximation algorithm with which we can adapt the hierarchical B-Spline space to the given data.

6.3 SUBDIVISION PROPERTY OF B-SPLINES AND NURBS

6.3.1 Uniform Basis

Now that the idea of the hierarchical basis is expressed, in a compact and solid way, it is necessary to introduce and present another basic ingredient of the local hierarchical refinement in isogeometric analysis, the subdivision property. Subdivision property is a unique feature of B-Splines and their generalization NURBS, which allows a random B-Spline of a certain level to be expressed as a linear combination of finer basis functions of more than one level. This procedure is recursively used in computational graphics in order to represent finer geometries with more details. Of course the right coefficients must be used in order to achieve the linear multi-level expression. For that purpose, two-scale relation (6.17) has been introduced by the researchers of computer graphics and geometry. For the following units we will consider the one-dimensional problem for B-Splines, but as it will be shown next, the generalization of the procedure is quite straightforward for the two or three dimensional cases.

$$N_{i,\mathrm{p}}^{\ell} = \sum_{k=0}^{p+1} S_{i,k}^{p} N_{2i+k,p}^{\ell+1}(\xi) \tag{6.17}$$

where

$$S_{i,k}^{p} = \frac{1}{2^{p}}\binom{p+1}{k} = \frac{1}{2^{p}}\frac{(\mathrm{p}+1)!}{(\mathrm{p}+1-k)!\mathrm{k}!} \tag{6.18}$$

k=0, 1, 2…p+1

Where p is the polynomial degree of basis functions and k is the necessary number of finer level basis functions that are needed to represent the coarse B-Spline. $S_{i,k}^{p}$ is called **subdivision (or coefficient) matrix** and it is a banded sparse non-square matrix. Its components are independent of the knot index i and the level. More specifically subdivision matrix has the following properties:

- It is a non-square matrix, where its rows correspond to the basis functions before refinement (maternal) while its columns to the refined basis functions (children). Their number is different. So it is a matrix with i rows and j columns.
- It is sparse. Each basis function at the coarser level has relations with at most p+2 children bases at the finer level.
- The elements in each column constitute a partition of unity. For $\forall j$ there is $\sum_{i=1}^{m} S_{ij} = 1$
- Each coefficient is non-negative and less or equal to 1.

We consider the above relation valid for a level ℓ function of a uniform knot vector, where knots are inserted in the region of the support of the selected B-Splines. Halving the corresponding knot spans, thus the number 2 is included in the relation. Furthermore, a generalization of the relation can occur for other equal subdivisions of the support such as three, four or five. However, splitting the existing knot spans into two smaller seems better for implementation in the computing of our algorithm, especially in higher dimension applications such as 2D and 3D, where the data must be handled efficiently. It is clear that other subdivision schemes would make the implementation more complicated, despite the fact that they would lead theoretically to an analysis suitable mesh.

Thus, the following relation occurs for the knot span length of the following levels. It is straightforward that through the different lengths all over the hierarchy of the basis, one can calculate the support of each basis function effortlessly.

$$\mathrm{h_k} = 2^{-\mathrm{k}} \cdot \mathrm{h},\ 1 \le \mathrm{k} \le \mathrm{K} \tag{6.19}$$

From the above relation, one can get also the relation between the supports of two or more consecutive levels.

$$\mathrm{supp}(\ell + \mathrm{m}) = \frac{1}{2^m}\mathrm{supp}(\ell) \tag{6.20}$$

Where m is the number of the examined refinement level, h is the initial length of the uniform case of knot vector and $\mathrm{h_k}$ is the length of the knot span for a random level k of the hierarchy. This way, the length can be calculated quickly.

For example:

- for k=0 $h_k = h$
- for k=1 $h_k = 0.5 \cdot h$
- for k=2 $h_k = 0.25 \cdot h$

where h can be directly derived from a knot vector $\{\xi_1, \xi_2, ..., \xi_i, ..., \xi_{n-1}, \xi_n\}$ such as,

$$h = \xi_{i+1} - \xi_i,\ i = 1, 2, ..., n-1 \tag{6.21}$$

This relation refers to a uniform knot value vector and subsequently to a uniform basis. The generalization of the two-scale relation for a non-uniform basis which takes into account the different knot span lengths as well as the reduced continuity or the knot multiplicity in certain knots will be presented later on.

An example will be now illustrated in **Fig. 6.6** in order to fully understand the nature of this relation and the result that can occur by using it.

Assuming we have a fully deployed B-Spline basis function that can be derived from the Cox-de Boor relation. The basis function has a knot vector of [0, 1, 2, 3] and degree p=2. No other functions are considered for this example, as our aim is to understand the mechanism behind the previous introduced relations.

In order to address and handle properly the multi-level relations, some terminology is required. Thus, we will from now consider two categories of basis functions for addressing the functions of two consecutive levels. First we will refer to the coarse basis functions that are selected to be refined and belong to level ℓ as maternal B-Spline functions, in the sense that they are the ones that can generate the finer level functions. Moreover the finer level ℓ +1 functions will be referred as subsidiary (children) B-Spline functions.

Now that the multi-level relations can be properly handled without being confused, we will use the algebraic equations.

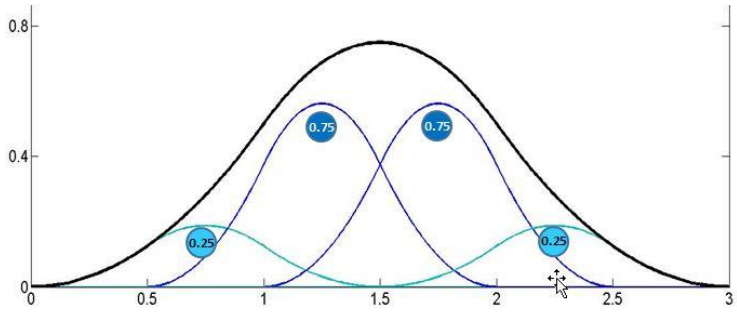


Figure 6.6.
The example's maternal function is shown with black color. The children with coefficients 0.25 and 0.75 are depicted with cyan and blue respectively
(Iakovos Antonios and Karras Dimitrios , 2015)

In **Fig. 6.6** we can see how the maternal function can be represented from *p+2* finer level functions. In addition the new knot vector is [0, 0.5, 1, 1.5, 2, 2.5, 3]. If we try to input the new knot vector to the Cox-de Boor relation 4 quadratic subsidiaries will emerge with the following length of support

$$\text{supp} = (\text{p}+1)\cdot \text{h}_k = (p+1)\cdot 0.5\cdot h \tag{6.22}$$

Where for p=2 and h=1 we get that every one of the subsidiary functions will have length of support equal to 1.5, which is the half of the maternal basis function.

However, the Cox-de Boor relation gives us the new subsidiary functions, but not their coefficients. In order to represent the maternal basis function by the set of subsidiaries we will use the subdivision property two-scale relation.

So from implementing the relation for p=2 and k=0 , the following p+2 coefficients occur, 0.25, 0.75, 0.75, 0.25 respectively for the four subsidiaries starting from left to right. As one can easily notice, the sum of the subsidiaries at every ξ that belongs in [0, 3] equals to the value of the maternal function.

The subdivision property is valid and thus can be applied for more than one level of refinement, by just following the hierarchy of the levels as well as input the correct parameters to the aforementioned relations.

Of course, the above theories apply for every polynomial degree p as displayed in **Fig. 6.7**.

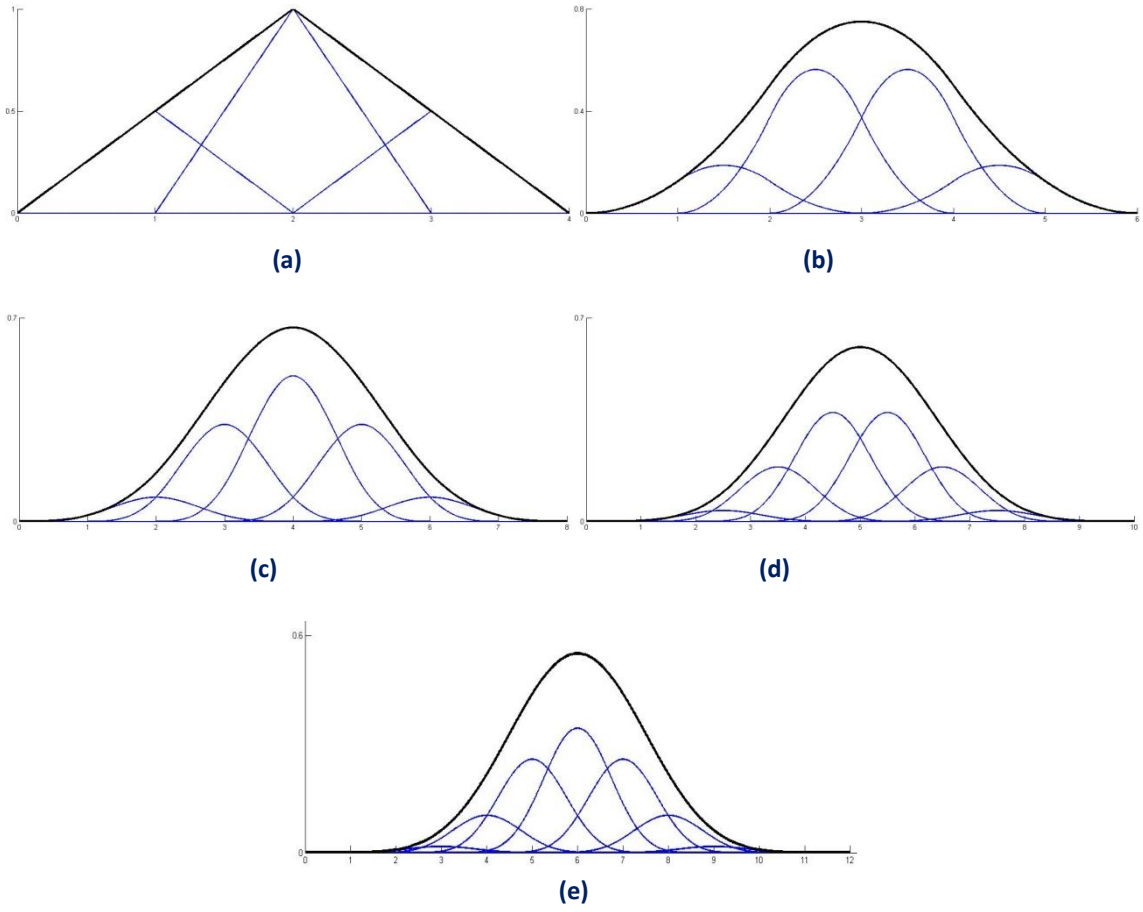


Figure 6.7.
B-Splines subdivision for **(a)** p=1, **(b)** p=2, **(c)** p=3 , **(d)** p=4 , **(e)** p=5 is illustrated. The number of children for each case is equal to (p+2)
(Iakovos Antonios and Karras Dimitrios , 2015)

In **Fig. 6.8.a**, the coefficients for two consecutive levels are depicted. More specifically these coefficients connect the finer functions of level 2, of contracted support with the coarse maternal B-Splines. By summarizing all the values of each column, multiplied by the coefficients of the previous level the coefficients of the univariate case for refinement level 2 can be derived.

In **Fig. 6.8.b**, the coefficients of a bivariate B-Spline of polynomial degree p=2 can be seen. More specifically these coefficients can be derived by applying the subdivision relation to each of the parametric local knot values and multiplying them in a full tensor way. So, the key factor in calculating coefficients in multivariate cases, is to derive the coefficients of each parametric axis independently. However this must not be confused with a global tensor product multiplication, as in h-refinement.

	N^3_1	N^3_2	N^3_3	N^3_4	N^3_5	N^3_6	N^3_7	N^3_8	N^3_9	N^3_{10}
0.25	0.25	0.75	0.75	0.25						
0.75			0.25	0.75	0.75	0.25				
0.75					0.25	0.75	0.75	0.25		
0.25							0.25	0.75	0.75	0.25
Final Weights	**0.0625**	**0.1875**	**0.375**	**0.625**	**0.75**	**0.75**	**0.625**	**0.375**	**0.1875**	**0.0625**

(a)

	0.0625	**0.1875**	**0.375**	**0.625**	**0.75**	**0.75**	**0.625**	**0.375**	**0.1875**	**0.0625**
0.0625	0.004	0.012	0.023	0.039	0.047	0.047	0.039	0.023	0.012	0.004
0.1875	0.012	0.035	0.070	0.117	0.141	0.141	0.117	0.070	0.035	0.012
0.375	0.023	0.070	0.141	0.234	0.281	0.281	0.234	0.141	0.070	0.023
0.625	0.039	0.117	0.234	0.391	0.469	0.469	0.391	0.234	0.117	0.039
0.75	0.047	0.141	0.281	0.469	0.563	0.563	0.469	0.281	0.141	0.047
0.75	0.047	0.141	0.281	0.469	0.563	0.563	0.469	0.281	0.141	0.047
0.625	0.039	0.117	0.234	0.391	0.469	0.469	0.391	0.234	0.117	0.039
0.375	0.023	0.070	0.141	0.234	0.281	0.281	0.234	0.141	0.070	0.023
0.1875	0.012	0.035	0.070	0.117	0.141	0.141	0.117	0.070	0.035	0.012
0.0625	0.004	0.012	0.023	0.039	0.047	0.047	0.039	0.023	0.012	0.004

(b)

Figure 6.8.
In this figure an easy calculation of the second consecutive level, subdivision coefficients is depicted. In **(a)** the recursive computation for 2 level of refinement is shown for the univariate case. In **(b)** the coefficients for a bivariate B-Spline of polynomial degree 2 is shown. The coefficients are produced by a tensor product multiplication of each of the parametric axes
(Iakovos Antonios and Karras Dimitrios , 2015)

6.3.2 Non-Uniform Basis

Now that we have investigated the uniform basis case, it is time to generalize the theory presented in the non-uniform case. The non-uniform case is not only characterized by different knot-span lengths, but also by cases of reduced continuity along the basis. This way a full set of necessary tools will be presented in order to handle properly every basis one may encounter

As the final goal is always programming, the presented knowledge may become easier interpreted by readers with the illustrative and analytical examples presented in the following pages. As we mentioned in the previous chapters, a basis might have scattered

knots or knot multiplicities. These cases are obtained through the CAD representation and most of the times are not up to the user to decide, unless he chooses to use multiple patches to handle complicated geometries.

In any case, as the patch interconnection is still a subject of study in NURBS and B-Splines (lack of watertight connections), in contrary with T-Splines, handling discontinuities or arbitrary bases is crucial. As presented before, a fully developed maternal basis can be described by exactly p+2 finer basis functions, by adjusting the right corresponding factors.

The same rules apply on the non-uniform cases for all the fully developed basis functions. A difference is spotted in the boundaries of a patch, where B-Splines are not fully developed and hence although they may be of degree p, they are described not by p+2 functions, but by only 2 in general. Of course in a theoretical approach, the subdivision property is valid, but for the non-uniform case much more effort is needed in order to acquire the right coefficients for every spline.

The multiplicity at the two end knots of an open knot vector is always p+1 and their related basis functions are discontinuous. These boundary basis functions have only two children basis functions and their coefficients are 1 and 0.5 respectively whatever their degrees are.

In contrast, the internal basis functions are associated with more children bases at the lower level. Their combinatorial coefficients not only depend on degree p but also the values of individual knots for non-uniform knot vectors. In detail, the refinement of basis functions with non-uniform knot vectors can be represented by the following formula.

$$N_{i,\mathrm{p}}^{\ell}(\xi) = [a][N] = \sum_{k=0}^{p+1} \alpha_j N_{i+k+p}^{\ell+1}(\xi) \tag{6.23}$$

For the quadratic basis function that is defined by a span $[\xi_1, \xi_2, \xi_3, \xi_4]$, the coefficients can be calculated using its local knot values:

$$\alpha_1 = \frac{\xi_2 - \xi_1}{2(\xi_3 - \xi_1)} \tag{6.24}$$

$$\alpha_2 = 1 - \frac{\xi_3 - \xi_2}{2(\xi_3 - \xi_1)} \tag{6.25}$$

$$\alpha_3 = 1 - \frac{\xi_3 - \xi_2}{2(\xi_4 - \xi_2)} \tag{6.26}$$

$$\alpha_4 = 1 - \frac{\xi_4 - \xi_3}{2(\xi_4 - \xi_2)} \tag{6.27}$$

- As for the cubic basis function with a local knot vector $\left[\xi_1,\xi_2,\xi_3,\xi_4,\xi_5\right]$ the coefficients can be identified in the similar way:

$$\alpha_1=\frac{\xi_2-\xi_1}{2\left(\xi_3-\xi_1\right)}\frac{\xi_3+\xi_2-2\xi_1}{2\left(\xi_4-\xi_1\right)} \tag{6.28}$$

$$\alpha_2=\frac{\xi_3+\xi_2-2\xi_1}{2\left(\xi_4-\xi_1\right)} \tag{6.29}$$

$$\alpha_3=\frac{\xi_4-\xi_3}{2\left(\xi_4-\xi_2\right)}\frac{-\xi_1+\xi_2+\xi_3-\xi_4}{2\left(\xi_4-\xi_1\right)}+\frac{\xi_3-\xi_2}{2\left(\xi_4-\xi_2\right)}\frac{\xi_2-\xi_3-\xi_4+\xi_5}{2\left(\xi_5-\xi_2\right)}+\frac{3}{4} \tag{6.30}$$

$$\alpha_4=\frac{-\xi_4-\xi_3+2\xi_5}{2\left(\xi_5-\xi_2\right)} \tag{6.31}$$

$$\alpha_5=\frac{+\xi_4+\xi_3-2\xi_5}{2\left(\xi_5-\xi_2\right)}\frac{\xi_5-\xi_4}{2\left(\xi_5-\xi_3\right)} \tag{6.32}$$

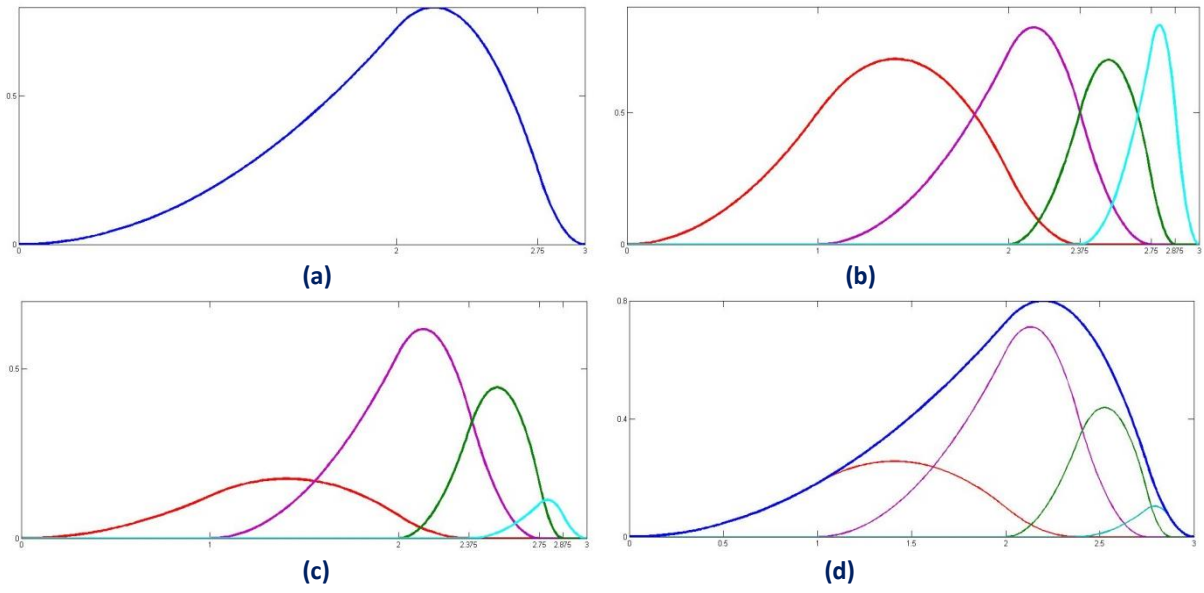


Figure 6.9.
Subdivision property in for a non-uniform B-Spline of polynomial degree p=3**. (a)** Maternal basis function. Course local knot vector [1 2.5 3.75 4.75 5.25] **(a)** Subsidiary basis functions. Refined bisected local knot vector [1 1.75 2.5 3.125 3.75 4.25 4.75 5 5.25] **(c)**Scaled subsidiary basis functions **(d)** Subdivision property representation
(Iakovos Antonios and Karras Dimitrios , 2015)

6.3.3 Subdivision of N-Variate B-Spline Shape Functions

After introducing and explaining in depth the subdivision property of B-Spline curves for the one dimensional case, we can proceed forward by generalizing the theory for the n-variate case of tensor product B-Splines.

6.3.3.1 2D Case

For the sake of simplicity and to ease notation and examples a bivariate B-Spline of degree p=2 is considered for both of the parameter axes ξ and η.

Let B be a tensor product B-Spline that is constructed by two random full developed one dimensional B-Splines of degree p=2 on ξ and η axes respectively. Each one dimensional basis function has a local knot vector and thus a support, as the following.

$[\xi_1, \xi_2, \xi_3, \xi_4]$, from which the $N_{i,2}^{\ell}(\xi)$ is derived and

$[\eta_1, \eta_2, \eta_3, \eta_4]$, from which $N_{j,2}^{\ell}(\eta)$ is derived as well

The corresponding tensor product B-Spline will be:

$$N_{i,\mathrm{j},2}^{\ell}(\xi,\eta) = N_{i,2}^{\ell}(\xi)\, N_{\mathrm{j},2}^{\ell}(\eta) \tag{6.33}$$

with support

$$\mathrm{supp}(\mathrm{N}_{i,j,2}^{\ell}) = [\xi_1, \xi_2, \xi_3, \xi_4]\, x\, [\eta_1, \eta_2, \eta_3, \eta_4] \tag{6.34}$$

From the above equations it is easy to understand that for degree p=q=2 for both axes, the tensor product bivariate basis function is comprised of $(p+1)^{n=2}$ cells, which consist its support.

As it was mentioned in the introduction chapter, hierarchical refinement is the technique of enriching the approximation space locally by replacing a coarse grid along with its correspondent basis functions, with finer ones. In this sense, knot insertion is performed in the local knot vectors in each axis ξ, η, ζ for each basis function separately.

By recalling the previous knowledge on local refinement that was presented, we know that for the one dimensional problem each maternal basis function has p+2 children (subsidiaries).

For the n-dimensional, here 2-dimensional problem, the number of children of the tensor product B-SPLine basis functions equals to

$$(p+2)(q+2) \text{ or } (p+2)^2 \tag{6.35}$$

for $p=q$, where p and q are the degree of ξ and η axes respectively.

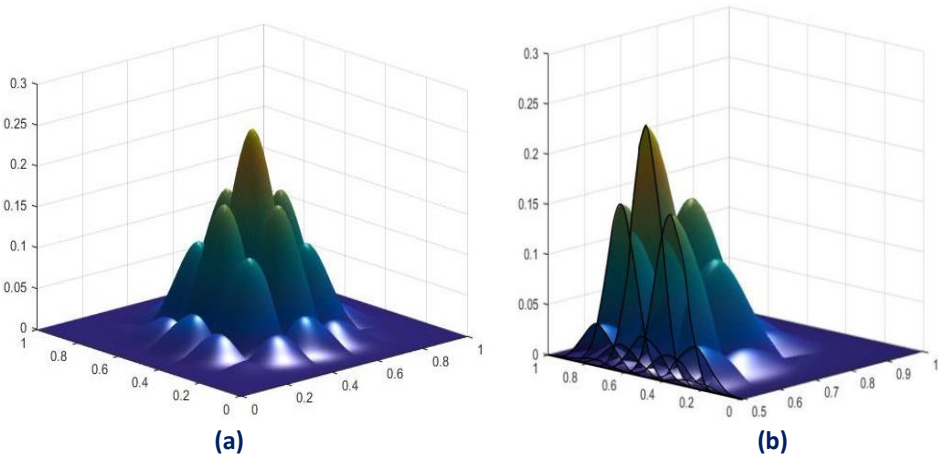


Figure 6.10.
(a) Graphical representation of a bivariate B-Spline's children **(b)** Cut along η axis

It is now time to find the coefficients for the tensor product children of the maternal basis function. By applying the previous subdivision relation for each of the axes separately and then combine them in a tensor product sense, the final coefficients can be derived. Consequently, due to their tensor product structure, the generalization of subdivision to multivariate B-Splines is a straightforward extension of the subdivision property relation and can be written as:

$$B^p(\zeta)=\sum_j\left(\prod_{\ell=1}^{d}2^{-p\ell}\left(\frac{p_\ell+1}{j_\ell}\right)N^{p\ell}(2\xi^\ell-j_\ell)\right) \tag{6.36}$$

where multi-indices,

- $j=\left\{i_1,\ldots,i_{d_p}\right\}$
- $p=\left\{p_1,\ldots,\mathrm{p}_{d_p}\right\}$
- $\zeta=\left\{\zeta^1,\ldots,\zeta^{d_p}\right\}$

denote the position in the tensor product structure, the polynomial degree and the independent variables in each direction ℓ of the n-dimensional parameter space. The use of subdivision property for the bivariate case is depicted in **Fig. 6.10**. Continuing, illustrated examples will be represented in order to completely understand the procedure of subdivision. The most widely known application of the aforementioned relations is the development of highly efficient subdivision algorithms for the fast and accurate approximation of smooth surfaces by control meshes in computer graphics.

6.3.3.2 Example for the coefficients of the 2D case B-SPLines

So in order to simplify the explanation, we will present an example to demonstrate how the subdivision coefficients are generated in the bivariate case for degree p=2 for both the parameter axes. We assume having a tensor product bivariate B-Spline with $(p+1)^2$ cells that consist its support in the 2D parameter space. When this particular spline is selected for refinement, the local knot vectors for both axes will change by dividing their corresponding knot spans in to half and along with them new smaller cells will be generated. By the knot insertion new children will be introduced with half maternal support.

In **Fig. 6.11**, the bivariate B-Spline is depicted, along with its random local knot vector on both parameter axes. After the knot insertion the initial local knot vectors,

$$\Xi=\left[\xi_1,\xi_2,\xi_3,\xi_4\right]$$

$$H=\left[\eta_1,\eta_2,\eta_3,\eta_4\right]$$

have become:

$$\Xi'=\left[\xi_1,\xi_5',\xi_2,\xi_6',\xi_3,\xi_7',\xi_4\right]$$

$$H'=\left[\eta_1,\eta_5',\eta_2,\eta_6',\eta_3,\eta_7',\eta_4\right]$$

where the knot values that are marked refer to the newly introduced knots on both axes.

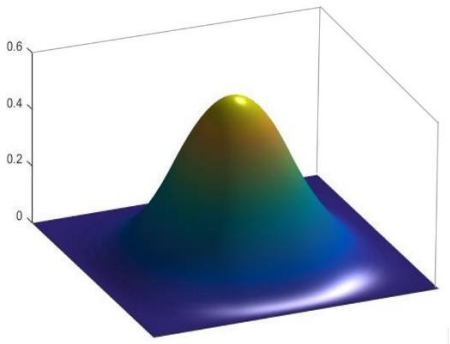


Figure 6.11.
Bivariate basis function.
(Iakovos Antonios 2015)

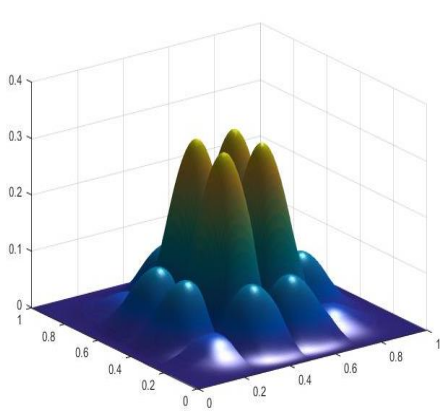
Now by applying the relation for the n-variate case of tensor product B-Spline curves for n=2, we can calculate the coefficients for the children that have been created during the subdivision of the maternal functions. Because of the tensor product nature of the maternal functions, the number of the children can be derived from equation (6.35).

In our case for p=q=2 the number of children equals to $(2+2)^2=16$ and its coefficients, in order to describe the maternal function, are as depicted in **Fig. 6.12.a**. Note that the outer rows and columns follow the same pattern. In addition to that, the center functions have

the highest coefficients, which is logical and it directly depends on the relation between the degrees of the two parameter axes.

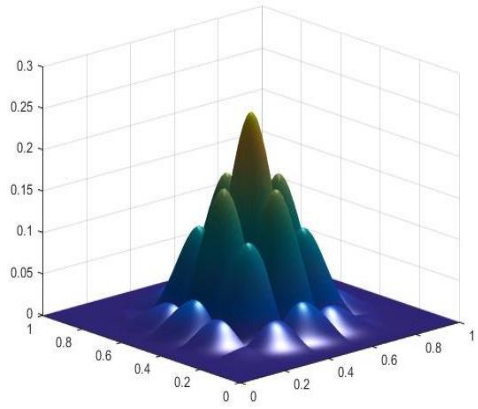
For simplicity reasons, although the maternal function has been presented in the parameter space, the children along with their coefficients will be represented in index space, due to the fact that B-Splines intersect and interact with p+1 other functions. This fact would lead to a confusing illustration of the parameter space.

In **Fig. 6.12.b** the case of cubic B-Splines is depicted. The number of children in this case is $(3+2)^2 = 25$. The scale factors pattern is similar to the one examined for the quadratic B-Splines.



	0.25	0.75	0.75	0.25
0.25	0.0625	0.1875	0.1875	0.0625
0.75	0.1875	0.5625	0.5625	0.1875
0.75	0.1875	0.5625	0.5625	0.1875
0.25	0.0625	0.1875	0.1875	0.0625

(a)



	0.125	0.5	0.75	0.5	0.125
0.125	0.016	0.063	0.094	0.063	0.016
0.5	0.063	0.250	0.375	0.250	0.063
0.75	0.094	0.375	0.563	0.375	0.094
0.5	0.063	0.250	0.375	0.250	0.063
0.125	0.016	0.063	0.094	0.063	0.016

(b)

Figure 6.12.
Scaled subsidiary shape functions. Scale factors have been produced from the subdivision property relation. **(a)** Quadratic B-Splines , **(b)** Cubic B-Splines.
(Iakovos Antonios 2015)

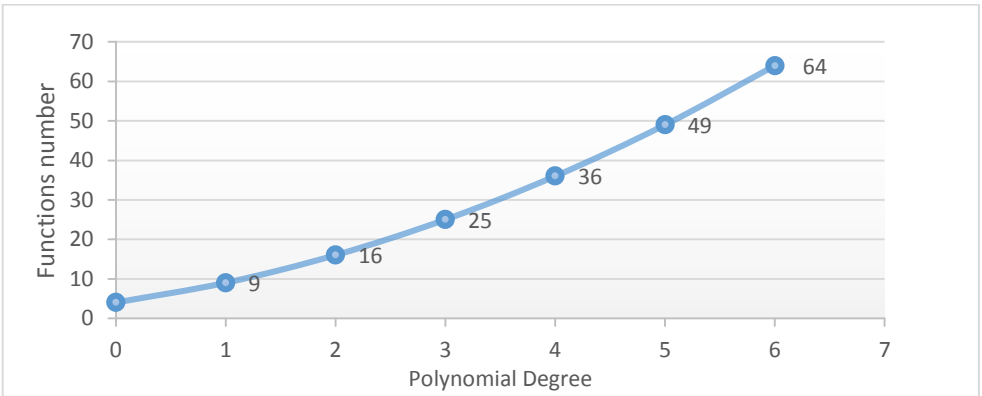


Figure 6.13.
Number of functions derived from 2D subdivision rule for one maternal shape function in relation to polynomial degree.

6.4 SUBDIVISION PROPERTY IN NURBS

As we already know, CAD programs do not use B-Splines in their algorithms. Instead they are using NURBS and thus the previous relations and proofs must be modified properly in order to be used in NURBS implementations.

It will be shown that, the generalization of the relations is easy and quite straightforward. In this case the study of B-Splines in the parameter and index space seems sufficient in order to achieve the desired results. However the modification of the related weights of each NURBS curve will be also presented for the sake of completeness. After proper handling of the relations for the one dimensional NURBS, the generalization for higher degrees will follow.

Subdivision rules for univariate NURBS derive by inserting the two-scale relation of the classical subdivision relation into the construction rule for NURBS basis functions, which eventually yields

$$R_{i,p}(\xi) = \frac{w_i \sum_{j=0}^{p+1} \left(\frac{p+1}{j} \right) N_{i,p}(2\xi - j)}{2^p \sum_{j=1}^{n} w_j N_{j,p}(\xi)} \tag{6.37}$$

According to the isogeometric framework, the geometry is described exactly by the original unrefined NURBS basis, so that geometry refinement in the same framework of isogeometric analysis is not required. Therefore, we keep the weight w_i and the corresponding control points P_j unchanged and always take the sum of the original B-Splines $N_{j,p}$ in the denominator of the above equation. Nonetheless, using the refined NURBS basis for enhancing the geometry representation would be of course possible.

A multivariate subdivision rule for NURBS can be derived analogously by substituting the classical rule for multi variable NURBS to the classical subdivision relation, which will give us the following expression

$$R^h_{i,p}(\zeta) = \frac{w_i \sum_{j=0}^{p+1} \prod_{\ell=1}^{d} 2^{-p_\ell} \left(\frac{p+1}{j_\ell} \right) N_{p_\ell}(2\xi^\ell - j_\ell)}{\sum_{j=1}^{n} w_j N_{j,p}(\zeta)} \tag{6.38}$$

Where the multi-index notation exactly follows the one introduced before for the multivariate B-Splines that were introduced in the previous chapters. Since the geometry is not refined the B-Splines $N_{j,p}$ and corresponding weights in the denominator of the above equation refer again to the original B-Spline patch.

6.5 HIERARCHICAL REFINEMENT FOR NURBS

On the basis of the previous subchapter, we introduce some modifications to the hierarchical refinement scheme for B-Splines to adapt the case of NURBS. First, the previous introduced relations are separated for the univariate and the bivariate case in two parts, a B-Spline part (numerator) and a rational part (denominator), which are treated separately. The numerator carries out hierarchical refinement on the B-Spline level, so that we can make full use of the concepts and the algorithms that will be later on introduced. Thus, the resulting refined NURBS basis is also constructed from a nested sequence of bisected knot span elements, over which multiple hierarchical overlay levels of repeatedly contracted B-Splines are defined.

As shown in the previous subchapter, the denominator is always computed with the original B-Spline basis $N^0_{j,p}(\sigma)$

$$sum(\xi) = \sum_j w_j N^0_{j,p}(\xi) \tag{6.39}$$

Where the multi-index *j* includes all B-Splines with support at the parameter space location *ζ*. The basis functions R_i of the hierarchical NURBS basis and its derivatives follow as

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi)}{sum(\xi)} \tag{6.40}$$

$$\frac{\partial R_{i,p}(\xi)}{\partial \xi^\ell} = \frac{\dfrac{\partial N_{i,p}(\xi)}{\partial \xi^\ell sum(\xi)} - N_{i,p}(\xi)\dfrac{\partial sum(\xi)}{\partial \xi^\ell}}{sum(\xi)^2} \tag{6.41}$$

Where we additionally drop the weights w_i in the numerator of the first equation for further simplification. Standard rules for generating higher order derivatives can be adapted in the same way, for use in colocation for instance. The geometry mapping by way of the Jacobian matrix and determinant are computed throughout the hierarchical refinement procedure form the unrefined NURBS basis

$$\chi(\xi) = \sum_j w_j \frac{N^0_{j,p}(\xi)}{sum(\xi)} P_j \tag{6.42}$$

with the initial set of weights w_j and control points P_j.

7. HIERARCHICAL REFINEMENT STRATEGIES

7.1 INTRODUCTION

The basic ingredients and tools needed in order to successively apply local hierarchical refinement in B-Splines and NURBS were presented in chapter 6. Basic terms such as the Kraft algorithm, nested domains, the subdivision property relation and extensive illustrated examples have been explained in order to comprehend in depth the rigorous mechanisms and equations used in the case of local refinement.

In the current chapter it is time to introduce, explain and eventually evaluate some method variations of hierarchical adaptive refinement introduced during the last few years. It has to been mentioned that the idea of adaptive refinement in isogeometric analysis is a relatively recent research topic. The first research ideas on hierarchical refinement have been publicated by Vuong et al in 2010. In their work they tried to implement R. Kraft's theory, presented in his PhD back in 1997, in the frame of isogeometric analysis. The idea emerged from the need for strictly local refinement in B-Splines and NURBS implementations in IGA, since the full tensor product nature seemed to be a major drawback for analysis purposes. An alternative approach to reach a successful and easy implementation of Vuong's theory was introduced by Bornemann et al. Later on, Giannelli came up with the truncated B-Spline concept, a method introducing new tools to optimize what was already proposed by Vuong. Another addition has been made by D. Schillinger by modifying existing methods in order to restrict refinement's area and strengthen its local nature. His proposals where reflected on the sparseness of stiffness and mass matrices. Just recently, a new kind of B-Splines, named LR B-Splines, has been introduced by Johannessen and Kvasmdal in order to achieve analysis suitable meshes and numerical results in local multi-level hierarchical refinement. It has to be clarified that all the aforementioned refinement strategies focus on the formation of hierarchical B-Spline and NURBS basis functions and as a result stiffness matrices.

All the aforementioned authors have set a very strong theoretical background for scientific society and they are still continuing on issuing papers which try to overcome the difficulties derived from hierarchical adaptive refinement. In this sense, research on local refinement in isogeometric analysis is still an ongoing issue. Many more ideas and proposals are made by various researchers in order to improve the existing theories and ease the implementation effort. In this thesis an algorithmic approach based on the previous knowledge will be

presented. The aim of this approach is to ease the procedures of the previous theories in IGA applications.

In the current chapter an extensive presentation will be made, so the reader will understand in depth the different refinement strategies and their advantages or disadvantages.

7.2 HIERARCHICAL REFINEMENT PROPOSED BY VUONG ET AL.

Refinement in general and adaptive local refinement in particular is one of the key issues in isogeometric analysis. Vuong et al. present an adaptive local refinement technique for IGA based on extensions of hierarchical B-Splines. The theoretical properties of the spline space are investigated, in order to ensure fundamental properties like linear independence and partition of unity. Furthermore, well established concepts from classical FEA are used to fully integrate hierarchical spline spaces into the isogeometric setting. Mesh refinement based on a-posteriori error estimators is a key ingredient for developing an effective isogeometric adaptive solver, as it will be described in more detail in the next unit. This is the main motivation for investigating possible extensions of tensor product splines, which provide an adaptive local control of the refinement, together with their suitable application into the analysis phase. Starting with an initial mesh grid, the error estimator iteratively identifies areas of the mesh, which require local refinement.

In addition to the aforementioned properties, some other definitions will be given. Having explained nested spaces extensively, it is time to move forward in introducing the method's fundamental theoretical principles.

Consider a finite sequence of N bivariate B-Spline spaces $(N^{\ell})_{\ell=0\ldots L-1}$ which are assumed to be nested and L is the number of hierarchical levels.

$$N^0 \subset N^1 \subset \ldots \subset N^{L-1} \tag{7.1}$$

Together with a finite sequence of N bounded open sets $(\Omega^{\ell})_{\ell=0\ldots L-1}$ with

$$\Omega^{L-1} \subseteq \Omega^{L-2} \subseteq \ldots \subseteq \Omega^0 \text{ and } \Omega^L = 0 \tag{7.2}$$

which define the nested domain for the spline hierarchy.

In order to obtain nested spaces, it is necessary to assume that

$$\mu\left(\Xi^{\ell+1},\xi\right) - \mu\left(\Xi^{\ell},\xi\right) \geq p^{\ell+1} - p^{\ell} \tag{7.3}$$

$$\mu\left(H^{\ell+1},\eta\right) - \mu\left(\mathrm{H}^{\ell},\eta\right) \geq q^{\ell+1} - q^{\ell} \tag{7.4}$$

$\forall$ ξ, η with $\ell = 0,\ldots,\mathrm{L}-2$.

These conditions are necessary and sufficient. It is usually considered that the polynomial degree remains constant for all levels of hierarchical refinement, an assumption also adopted by this thesis.

At each level, the boundary $\partial\Omega^{\ell}$, $\ell = 0,\ldots,\mathrm{L}-1$, may be aligned with the knot lines of $N^{\ell-1}$ which will be referred as **strong condition** or B^{ℓ} which will be referred as **weak condition**.

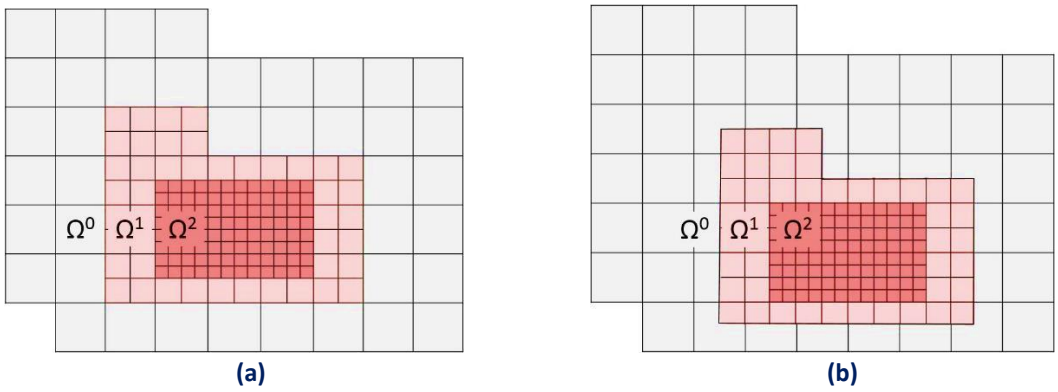


Figure 7.1.
(a) Strong boundary conditions between the subdomains**. (b)** Weak boundary conditions between the subdomains.
(Vuong et al, 2011)

Fig. 7.1 depicts the strong and weak boundary conditions between the nested subdomains. The initial domain Ω^0 is an arbitrary 2D parameter space. The refined domain Ω^1 is produced when the first refinement is conducted, while its parametric cells have the half side dimension in comparison with the initial domain Ω^0. Similarly, the refined domain Ω^2 is produced when the second refinement is conducted, while its parametric cells have the half side size in comparison with the domain Ω^1, as the subdivision number is chosen to be equal to 2.

It is worth mentioning that for the strong boundary conditions between the subdomains the boundary $\partial\Omega^1$ is aligned with the already existed knot lines of the initial domain Ω^0. On the other hand, this is not valid for the case of weak boundary conditions. The same pattern is noticed for the next refinement level. For the algorithmic point of view, the strong condition is considered better and more robust than the weak one. The case encountered in this thesis is strictly the one of strong boundary conditions between subdomains.

In hierarchical refinement, although the linear independence is always secured, partition of unity may be not satisfied directly along the basis. For this reason, we borrow the terminology used in T-Spline implementations, in order to classify the different hierarchical bases by the following definition.

B-Spline basis function $\{N_i\}$ will be referred as:

- **Standard** if $\sum N_i = 1$ (7.5)
- **Semi-Standard** if there exists a choice of weights $S_i > 0$, such that $\sum s_i N_i = 1$ (7.6)
- **Non-Standard** $\nexists$ no choice of $S_i > 0$ to ensure in any way the partition of unity (7.7)

In the general case of hierarchical B-Spline basis, the existence of all aforementioned definitions is allowed. A tensor product mesh will yield a standard basis, as we already know, but an arbitrary mesh will not have the same behavior. The ultimate goal is to always define an admissible mesh for analysis, which will at least be a semi-standard one.

Examples of standard and semi-standard bases are depicted in **Fig. 7.2**.

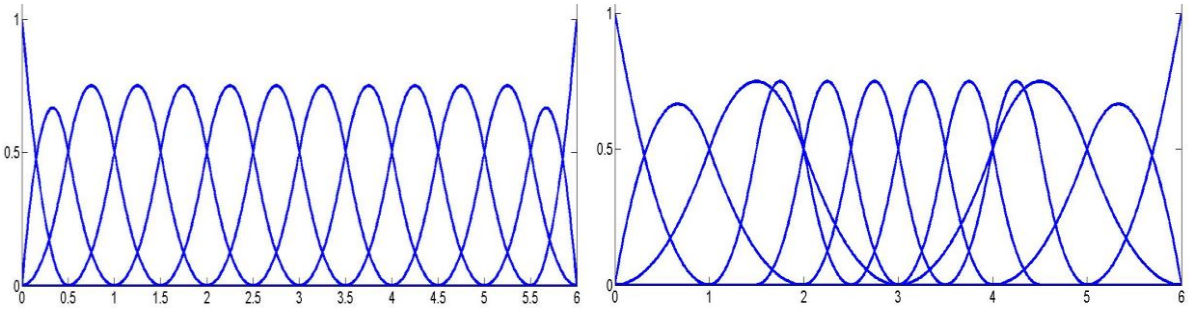


Figure 7.2.
(a) Standard basis. **(b)** Semi-Standard basis.
(Iakovos Antonios, 2015)

7.1.1 Constructing the hierarchical basis in a multi-level case

Now a generic algorithmic relation will be presented, that can be applied recursively between two consecutive levels of the hierarchy in order to obtain the hierarchical basis and inherit its desired properties.

The hierarchical spline space is recursively constructed as described below:

I. Initialization:

$$H^0 = \left\{ N_i \in \mathrm{N}^0 : \operatorname{supp} N_i \neq 0 \right\} \tag{7.7}$$

II. Recursive Case:

$$H^{\ell+1} = H_A^{\ell+1} \cup H_B^{\ell+1},\ for\ \ell = 0,\ldots,\mathrm{L}-1 \tag{7.8}$$

Where

$$H_A^{\ell+1} = \left\{ N_i \in N \in H^{\ell}, \operatorname{supp} N_i \subsetneq \Omega^{\ell+1} \right\} \tag{7.9}$$

$$H_B^{\ell+1} = \left\{ N_i : N_i \in \mathrm{N}^{\ell+1}, \operatorname{supp} N \subseteq \Omega^{\ell+1} \right\} \tag{7.10}$$

Where M is the number of the total levels.

III. Final step

$$H \equiv H^L \tag{7.11}$$

In step (I), the hierarchical basis is initialized, by adding all the functions of the coarse domain. This happens always for normal parameter spaces, but may differ for arbitrary ones. The area or areas of interest are selected by estimating their corresponding error. The

union of these areas forms the domain refinement will take place. The refined domains will become smaller and smaller, as the number of iterations increases due to the nested function domain property.

Now that the first domain Ω^{ℓ} is constructed, the algorithm proceeds to step (II). In this step, the algorithm identifies which functions have the necessary properties in order to be included to the hierarchical basis. Two distinct groups of basis functions are constructed. Group (A) includes the functions whose support is not fully contained in the next level's finer domain Ω^{l+1} and thus will remain intact. Group (B) consists only by functions of the finer level. From those functions only the ones with a support belonging completely to the next levels domain Ω^{l+1} are kept.

Note that functions contained in group (B) and are consequently included in the hierarchical basis of level ℓ, may be discarded in the next iteration step if their full support is part of the refined domain Ω^{l+2}.

To summarize, the recursive definition ensures that the algorithm always selects to include in the hierarchical basis the appropriate functions. The first step initializes the hierarchical basis with all the relevant functions of the underlying tensor product basis N^0.The recursive procedure then updates the basis by removing the coarse functions contained inside the refined region and including finer ones by substituting them.

As a result of the aforementioned definitions and the recursive algorithm, the hierarchical B-Spline basis and all the associated spaces have the following important properties, which will be proved later on.

- The functions of the hierarchical basis are linearly independent.
- The spaces spanned by the basis are nested, in a sense that $H^{\ell} \subseteq spanH^{\ell+1}$

7.1.2 Proof of linear independence

We just have to prove that $\sum_{N_i \in H} \alpha_i N_i = 0 \Rightarrow \alpha_i = 0$. This sum can be re-arranged as:

$$\sum_{N_i \in H \cap \mathrm{N}^0} a_i N_i + \sum_{N_i \in H \cap \mathrm{N}^1} a_i N_i + \ldots + \sum_{N_i \in H \cap \mathrm{N}^{L-1}} a_i N_i = 0 \qquad (7.12)$$

Since the functions in $H \cap \mathrm{N}^0$ are a subset of N^0, they are linearly independent as we have proved in the previous chapters. Only these functions are non-zero on $\Omega^0 \setminus \Omega^1$, hence in view of their local linear independence, we conclude that $a_i = 0$ for $\tau \in H \cap \mathrm{N}^0$.

Analogously when we consider all next sums in the sequence, we may observe that for each $\ell = 1,2,...,\mathrm{L}-1$ the functions in $H \cap \mathrm{N}^{\ell}$ are linearly independent. Except for functions already considered in the previous sums, namely in $H \cap \mathrm{N}^{0} ... H \cap \mathrm{N}^{\ell-1}$, only the functions in $H \cap \mathrm{N}$ are non-zero on $\Omega^{\ell} \setminus \Omega^{\ell+1}$. This implies that $a_i = 0$ for $\tau \in H \cap \mathrm{N}^{\ell}$ with $\ell = 1,2,...,\mathrm{L}-1$.

7.1.3 Proof of nested spaces

Lemma: Let $H^0, H^1, ..., H^{L-1}$ be the spline bases considered from the iterative procedure. Then the following is valid:

$$spanH^{\ell} \subseteq spanH^{\ell-1} \text{ For } \ell = 0,1,...,L-2$$

It will be proved that every function $C(\xi) \in spanH^{\ell}$ with $\ell = 0,1,...,L-2$ can be written as:

$$C(\xi) = \sum_{N_i \in H^{\ell}} a_i N_i = \sum_{N_i \in H^{\ell}, \mathrm{supp}\mathrm{N}_i \not\subset \Omega^{\ell+1}} a_i N_i + \sum_{N_i \in H^{\ell}, \mathrm{supp}\mathrm{N}_i \subseteq \Omega^{\ell+1}} a_i N_i \qquad (7.13)$$

where the first sum in the right-hand side of (7.14) collects all basis functions in $H^{\ell+1}$ (7.10). In view of the nested nature of the underlying spaces $B^0,...,B^{L-1}$, each function $N_i \in N^{\ell}$ can be expressed as linear combination of basis functions which belong to $N^{\ell+1}$, such that:

$$N_i^{\ell} = \sum_{N_j^{\ell+1} \in N^{\ell+1}, \mathrm{supp}\, N_j^{\ell+1} \subseteq \mathrm{supp}\, N_i^{\ell}} a_{i,(N_i^{\ell})}^{\ell+1} \cdot N_j^{\ell+1} \qquad (7.14)$$

where we denote by $a_{i,(N_i^{\ell})}^{\ell+1}$ the coefficient of $N_j^{\ell+1}$ in this expansion of N_i^{ℓ}. By substituting the above relation for N_i^{ℓ} into the second sum in (7.14), the following is derived:

$$C(\xi) = \sum_{N_i \in H_A^{\ell+1}} a_i N_i + \sum_{N_i \in H^{\ell}, \mathrm{supp}\mathrm{N}_i \subseteq \Omega^{\ell+1}} a_i \left(\sum_{N_j^{\ell+1} \in N^{\ell+1}, \mathrm{supp}\, N_j^{\ell+1} \subseteq \mathrm{supp}\, N_i} a_{i,(N_i^{\ell})}^{\ell+1} N_j^{\ell+1} \right) \qquad (7.15)$$

Since $\mathrm{supp}\, N_j^{\ell+1} \subseteq \mathrm{supp}\mathrm{N}_i^{\ell} \subseteq \Omega^{\ell+1}$ and observing that additional coefficients of basis functions $N_j^{\ell+1}$, whose support is not contained in the support of the function N_i^{ℓ} considered in (7.15) are zero, it can be considered $\mathrm{supp}\, N_j^{\ell+1} \subseteq \Omega^{\ell+1}$ and swapping the order of the two rightmost sums on the above equations leads to:

$$C(\xi) = \sum_{N_i \in H_A^{\ell+1}} a_i N_i + \sum_{N_j^{\ell+1} \in N^{\ell+1}, \mathrm{supp}\, N_j^{\ell+1} \subseteq \mathrm{supp}\, N_i} \left(\sum_{N_i \in H^{\ell}, \mathrm{supp}\mathrm{N}_i \subseteq \Omega^{\ell+1}} a_i \cdot a_{i,(N_i^{\ell})}^{\ell+1} \right) N_j^{\ell+1} \qquad (7.16)$$

which can be written more simply as:

$$\sum_{N_i \in H_A^{\ell+1}} a_i N_i + \sum b_j N_j^{\ell+1} \quad \text{where} \qquad b_j = \sum_{N_i \in H^{\ell}, \text{suppN}_i \subseteq \Omega^{\ell+1}} a_i \cdot a_{i,(N_i^{\ell})}^{\ell+1} \tag{7.17}$$

The first sum of (7.18) refers to $H_A^{\ell+1}$ and the second to $H_B^{\ell+1}$. So, $C(\xi) \in spanH^{\ell+1}$.

7.1.4 A weighted hierarchical basis

The key property of B-Spline representations is that the related curve or surface is a convex combination of the underlying control points. In order to achieve this, within the framework of the hierarchical approach, hierarchical basis H may be modified to obtain the same properties with basis H_M. One such property is partition of unity. Moreover, since B-Spline functions are non-negative, by defining hierarchical spline basis functions the curve/ surface/ solid will be a convex combination of its control points, implying the convex hull property.

If the constant function belongs to the span of H^0, we can always represent it by the hierarchical basis H as:

$$\sum_{N_i \in H} s_{N_i} N_i = 1 \tag{7.18}$$

Hence, if for each basis function N_i that belongs to *H*, we define a related weighted basis function $\Psi_i = s_{N_i} N_i$, assuming that $s_{N_i} \neq 0$, we obtain the normalized hierarchical basis.

$$\mathrm{H_M} = \left\{ \Psi_i = s_{N_i} N_i : N_i \in H \wedge 1 = \sum_{N_i \in H} s_{N_i} N_i \right\} \tag{7.19}$$

which will ultimately form a partition of unity, as follows

$$\sum_{\Psi_i \in \mathrm{H_M}} \Psi_i = 1 \tag{7.20}$$

Since $\mathrm{N}_i^0 \in spanH^0$ it can be written as:

$$\sum_{N_i \in N^0} s_{N_i} N_i = 1 \tag{7.21}$$

with each $s_{N_i} \geq 0$ and then

$$\sum_{N_i \in N^0, \text{supp}\, N_i \not\subset \Omega^1} s_{N_i} N_i + \sum_{N_i \in N^0, \text{supp}\, N_i \subset \Omega^1} s_{N_i} N_i = 1 \tag{7.22}$$

The first sum in the right-hand side of the above relation collects all basis functions in H_A^1. Also, each basis function $N_i \in \mathrm{N}^0$ can be expressed as a linear combination of finer level basis functions, which belong to N^1 as it has already been proven, so:

$$N_i^\ell = \sum_{N_j^{\ell+1} \in N^{\ell+1}, \text{supp}\, N_j^{\ell+1} \subseteq \text{supp}\, N_i^\ell} a_{i,(N_i^\ell)}^{\ell+1} \cdot N_j^{\ell+1}, \text{ with } a_{i,(N_i^\ell)}^{\ell+1} \geq 0.$$

By substituting (7.19) to (7.18), the following relation is derived as:

$$\sum_{N_i^\ell \in \mathrm{H}_\mathrm{A}^1} a_{N_i^\ell} N_i^\ell + \sum_{N_i^\ell \in N^0, \text{supp}\, N_i^\ell \subset \Omega^1} a_{N_i^\ell} \left(\sum_{N_j^{\ell+1} \in \mathrm{N}^1, \text{supp}\, N_j^{\ell+1} \subseteq \Omega^1} a_{i,(N_i^\ell)}^{\ell+1} N_j^{\ell+1} \right) = 1 \tag{7.23}$$

Swapping the order of the two rightmost sums leads to the following:

$$\sum_{N_i^\ell \in \mathrm{H}_\mathrm{A}^1} a_{N_i^\ell} N_i^\ell + \sum_{N_j^{\ell+1} \in \mathrm{N}^1, \text{supp}\, N_j^{\ell+1} \subseteq \Omega^1} \left(\sum_{N_i^\ell \in N^0, \text{supp}\, N_i^\ell \subset \Omega^1} a_{N_i^\ell} a_{i,(N_i^\ell)}^{\ell+1} \right) N_j^{\ell+1} = 1 \tag{7.24}$$

$$\sum_{N_i^\ell \in \mathrm{H}_\mathrm{A}^1} a_{N_i^\ell} N_i^\ell + \sum_{N_j^{\ell+1} \in \mathrm{H}_\mathrm{B}^1} b_j N_j^{\ell+1}$$

$$\text{where } b_j = \sum_{N_i^\ell \in N^0, \text{supp}\, \gamma \subset \Omega^1} a_{N_i^\ell} \cdot a_{i,(N_i^\ell)}^{\ell+1} \geq 0$$

7.1.5 Active B-Spline basis functions

The set of active basis functions H is defined as follows:

A function $N_i \in N^\ell$ of level ℓ is an element of H if

$$\text{supp}\, N_i \subset \bar{\Omega}^\ell \tag{7.25}$$

and

$$\text{supp}\, N_i \not\subset \bar{\Omega}^{\ell+1} \tag{7.26}$$

This means, that a function is active on a level ℓ, when its support belongs completely to the level domain, without intersecting with coarser domains. The active functions coincide with the definition of the hierarchical basis and thus the use of the same identifier procedure as in H is justified. In addition, the properties proved earlier in this chapter hold as well: the active basis functions are linearly independent and the original space is a subset of the refined space defined by the active basis functions.

$$\mathrm{H}^0 \subseteq spanH \tag{7.27}$$

This implies that it is still possible to represent the exact geometry in the refined space. Partition of unity can be also preserved with use of scale factors as proved before. Summarizing, all the essential properties needed for using the basis functions in isogeometric analysis are ensured by construction.

An example of active basis functions on three consecutive levels is shown in **Fig. 7.3** below.

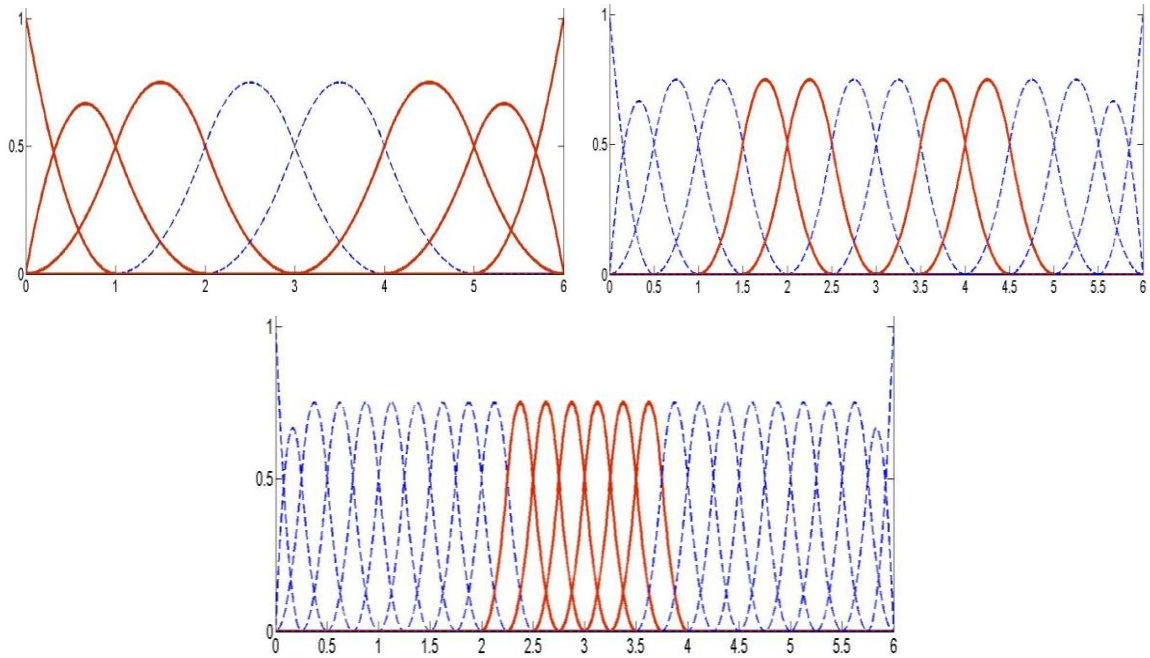


Figure 7.3.
Three consecutive hierarchical refinement levels.
Active basis functions on each level are depicted with red color.
(Iakovos Antonios and Dimitrios Karras, 2015)

7.2 THE BORNEMANN-CIRAK IMPLEMENTATION PROPOSAL

7.2.1 Brief Overview

In 2012, Bornemann and Cirak published their work on classical hierarchical refinement. The two researchers have continued the work of Vuong et al. They introduced a fast and accurate method in order to calculate the necessary coefficients Vuong proposed to overcome the possible lack of partition of unity property.

They tried to introduce a strategy in an algorithmic sense easy to compute and even easier to handle through the different levels of the hierarchy. Their implementations focus more on B-Spline applications. However, it is clearly mentioned that the generalization and therefore the extension of their proposal to NURBS cases might be possible.

Their proposal based on Vuong's theoretical heritage is enriched with subdivision property of B-Splines in order to acquire coefficients that are in good agreement between hierarchical levels. In this unit it will be considered that Vuong's proposal is valid with all its proofs and lemmas.

7.2.2 Coefficients handling

As we already know the s_i factors are referring to the hierarchical basis and more specifically to the B-Splines where the evaluation takes place. Instead of applying those corrective scale factors to basis functions it is possible to define control point coefficients by applying them directly to the control points. Of course all the calculations are executed by considering the data and characteristics of the basis.

In this sense, a spline on level ℓ is defined as the product of the basis function of this level N_i^ℓ with the coefficients (control points):

$$u(\xi) = \sum N_i^\ell(\xi)\,\mathrm{u}_i^\ell = N^\ell u^\ell \tag{7.28}$$

And the same spline can be represented in the next finer level ℓ+1:

$$u(\xi) = \left(S \cdot N^{\ell+1}\right) \cdot u^\ell = N^{\ell+1} \cdot \left(S^T u^\ell\right) \tag{7.29}$$

From (7.30) follows that the coefficients of the refined level are given with the subdivision relation:

$$u^{\ell+1} = S^T \cdot u^\ell \tag{7.30}$$

The same relations are valid for the n-variate cases and the generalization is actually quite simple. In this case, the subdivision matrix S and its coefficients are constructed as the tensor product of univariate subdivision weights. So, 3D subdivision matrix is given by the following expression:

$$S^p = S_i^p = S_{i,\xi}^p \cdot S_{i,\eta}^p \cdot S_{i,\zeta}^p \tag{7.31}$$

The important fact about these relations and expressions is that they manage to represent exactly the same geometry as the coarse mesh allowing to retrieve more precise analysis results.

7.2.3 Identifying B-Splines

In order to perform hierarchical refinement it is very important to manipulate the data connecting different hierarchical levels. Such data is the identity of functions, their interactions and the grid cells these functions “intersect” with. The task becomes even more difficult by introducing finer functions and removing courser ones. It is easy to comprehend that as the dimensions of the examined problem are increasing, the complexity increases exponentially.

For the success of the identification process, proper indices must be determined. These indices must have a unique one way relation with the corresponding functions of the mesh. Two ways to address them are considered possible. The first way suggests that each function will be accompanied by its parametric coordinates. This means that one index is needed for the 1D case and two or three indices for the 2D and 3D cases respectively. The second way is based on the idea of addressing a single parametric global number to each function, a parametric function ID.

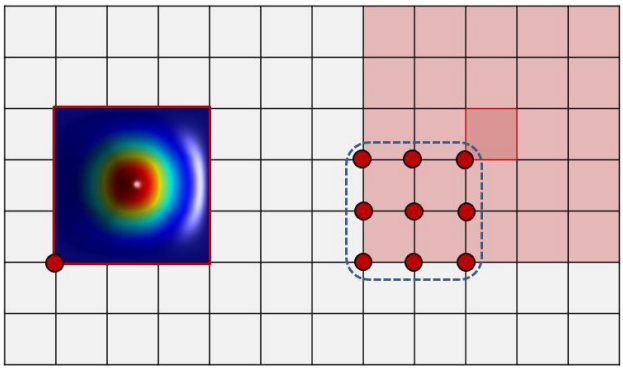


Figure 7.4.
Identification of the B-Splines with non-zero support over a 2D element (integration cell) as proposed by Bornemann. 2D B-Spline functions are identified by their lower left knot. The red dots are reference points of B-Spline functions that have non-zero values over the highlighted element. The colored area represents their total support
(Antonios Iakovos, 2015)

In order tackle this issue, Bornemann et al introduced the set $\mathfrak{I}_j^\ell$, which collects the B-Splines that belong to a certain level ℓ, whose supports have non-empty intersection with a particular integration (grid) cell and thus are non-zero in this cell such that:

$$\mathfrak{I}_j^\ell = \left\{ i \in \mathbb{Z}^d \mid \square_j^\ell \subset \mathrm{supp} \mathrm{N}_{i,p}^\ell \right\} = \{ j-p, \ldots, \mathrm{j} \} \tag{7.32}$$

The identification of 2D basis functions as proposed by Bornemann is depicted **Fig. 7.4**.

7.2.4 Refinement procedure

Now that necessary terms and mechanisms have been introduced the next step is to proceed with the explanation of the refinement procedure. In general, this proposal consists of three basic steps referred as:

- **Refinement**, which introduces new finer B-Splines in each level.
- **Compilation**, which assures the linear independence, by removing the redundant basis functions.
- **Shrinkage**, which discards B-Splines at the boundaries, of the area of interest, with a support not intersecting with the effective domain.

Assume that some shape functions are chosen to be refined by replacing them with finer functions. In the area of their support a denser grid is produced as a result of their refinement. It is known that the union of supports of the finer functions constitutes a domain nested in the coarser one. The 2D case of this procedure is shown in **Fig. 7.5**.

However, the newly introduced finer functions are not linearly independent in general. So, the aim of compilation step is to assemble a linearly independent basis by systematically discarding redundant B-Splines. A key issue for the optimal handling of the functions is to start from the maternal level and proceed by respecting the sequence of hierarchical levels in order to simplify the procedure by using relations connecting two consecutive levels. A coarse B-Spline has to be discarded if it can be represented as linear combination of existing finer functions. Compilation procedure stops at level L-1.

To summarize B-Splines of level ℓ , B_i^ℓ have to be discarded when:

$$\mathrm{supp}\, B_i^\ell \subset \Omega^{\ell+1} \tag{7.33}$$

After discarding the non-needed functions of each level, we can recall the step (II) of Vuong's recursive algorithm in order to construct our hierarchical basis as shown in **Figure 7.3**.

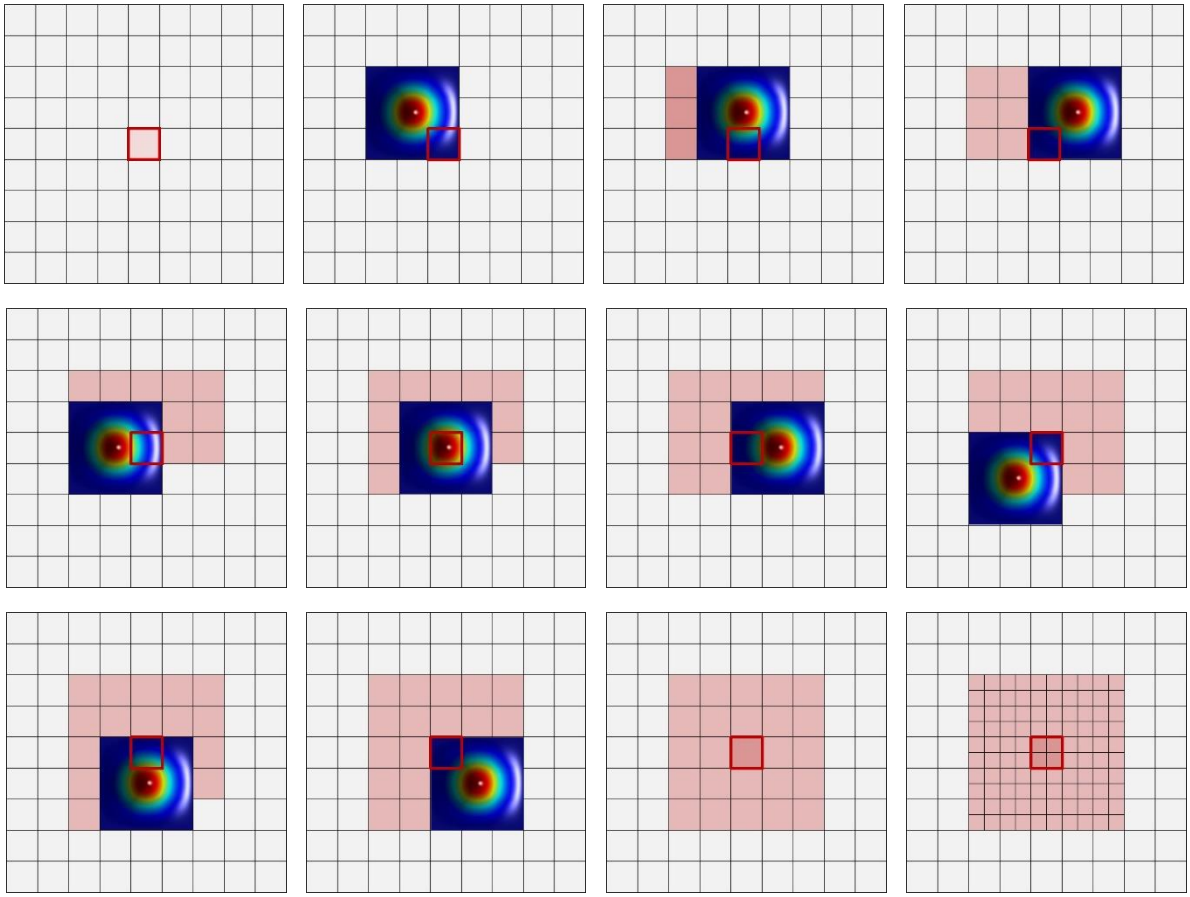


Figure 7.5.
Let assume we chose to refine the orange colored 2D element in the top left corner. This is a 2D tensor product space of polynomial degree p=2. Thus, for every coarse element 9 functions have non-zero support. Consequently, the intersecting functions must be refined. This leads to a refined area larger than the wanted one. Unfortunately this is a feature characterizing most of the proposed refinement strategies.
(Antonios Iakovos, 2015)

Handling efficiently the boundaries between the levels of the hierarchy seems to be a key issue in order to have an analysis suitable mesh. As it was previously mentioned, the H-basis is incomplete in grid cells close to the domain boundary of the previous level. Its polynomial reproduction properties are compromised, because some functions are missing in certain boundary cells (from the compilation step). It is straightforward that only the cells, that are p (polynomial degree) cells away from the boundary of the domain, have the full set of non-zero B-Splines.

For this purpose, Bornemann et al have introduced a new domain entity, named effective domain Ω. This domain is comprised of cells, which have the complete set of non-zero B-Splines. According to the authors, the cells between the effective and initial domains may be regarded as ghost cells. An example of this effective domain is presented in **Fig. 7.6**.

The effective domain can be determined, by counting the intersecting B-Splines of the cells on the initial (previous in general) level. If the number of B-Splines in:

$$P_i^0 = \left\{ i \in A_c^0 \mid \Omega_i^0 \subset \mathrm{supp}\mathrm{N}_i^0 \right\} \tag{7.34}$$

on level 0 is equal to $(p+1)^d$, where d is the number of dimensions, then a complete basis is available on the particular cell. The union of all cells with complete basis forms the effective domain such that:

$$\Omega = \bigcup_{i \in O^0} \Omega_i^0 \text{ where } O^0 = \left\{ i \in \mathbb{Z}^d \mid \left| P_i^0 \right| = (p+1)^d \right\} \quad (7.35)$$

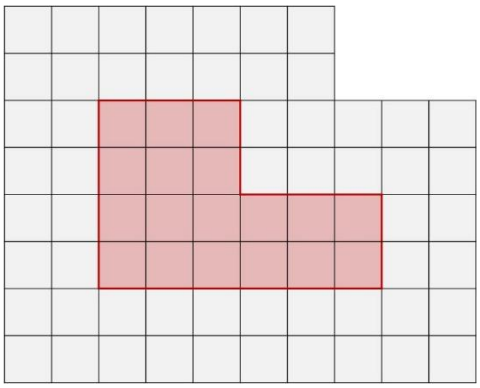


Figure 7.6.
Initial and effective domain in case of quadratic B-Splines (p=2). The effective domain is the union of the shaded cells
(Antonios Iakovos, 2015)

By taking into account all the previous notations and terms, we can express the definition for active B-Splines. So, the index set A_c^ℓ for the active B-Splines on a level ℓ occurs from the following three conditions in each domain:

$$A_c^\ell = \mathrm{i} \in \mathbb{Z}^d \mid$$

$$\left\{ (\operatorname{supp} N_i^\ell \cap \Omega) \subseteq \Omega^\ell \right\} \quad (7.36).$$

and

$$\left\{ (\operatorname{supp} N_i^\ell \cap \Omega) \not\subset \Omega^{\ell+1} \right\} \quad (7.37)$$

and

$$\left\{ (\operatorname{supp} N_i^\ell \cap \Omega) \neq \varnothing \right\} \quad (7.38)$$

..The three conditions in the index set A_c^ℓ reflect the procedures of compilation, refinement and shrinkage. These conditions are complemented with the property of nested subdomains. The hierarchical tensor product B-Splines on the effective domain form the space:

$$H_{(\Omega)} := span\{ N_i^\ell \mid 0 \le \ell \le \ell_{\max} \} \text{ where } i \in A_c^\ell \quad (7.39)$$

7.2.5 Subdivision Property of HB-Splines

By recursively applying the procedures applied for levels 0 and 1, the generalization can be easily obtained for any two levels ℓ and ℓ+1. To do this, a formal procedure is needed, in order to inherit the properties of subdivision property.

So, a random Spline C(ξ) on level 0 can be described as:

$$C(\xi) = \sum_{i \in N^0} N_i^0(\xi) \cdot u_i^0 = N^0(\xi) \cdot u^0 \tag{7.40}$$

The random Spline can be interpolated with hierarchical B-Splines from the space $H_{(\Omega)}$:

$$C(\xi) = \sum_{\ell} \sum_{i \in A_c^\ell} N_{i,p}^\ell(\xi) \cdot u_i^\ell = \sum_{\ell} N_H^\ell(\xi) \cdot \mathrm{u}_H^\ell \tag{7.41}$$

where the vectors N_H^ℓ and u_H^ℓ contain the active B-Splines and their coefficients on level ℓ respectively. Equations (7.41), (7.42) represent the same spline, only when coefficients u_H^ℓ are determined with a refinement relation. More specifically, these coefficients are computed by recalling the classical subdivision property relation. This relation is selectively applied to the coefficients of inactive (redundant) B-Splines. So, (7.41) can be rewritten as:

$$C(\xi) = \mathrm{N}_H^0(\xi) \cdot \mathrm{u}_H^0 + \sum_{i \in A_c^0} N_i^0(\xi) \cdot u_i^0 \tag{7.42}$$

With the second term, containing the inactive B-Splines on level 0. The coefficients of the inactive B-Splines are transferred to the B-Splines on level 1 using the subdivision relation.
For example:

$$u_j^1 = \sum_{i \in A_c^0} \left\{ S_{ij-2i}^p \cdot u_i^0 \right\} \tag{7.43}$$

The coefficients u_j^1 comprise the coefficients of inactive and active HB-Splines on level 1. After extracting the coefficients u_H^1 of the active B-Splines N_H^1 on level 1, C (ξ) can be rewritten as:

$$C(\xi) = N_H^0(\xi) \cdot \mathrm{u}_H^0 + N_H^1(\xi) \cdot \mathrm{u}_H^1 + \sum_{i \in A_c^1} N_i^1(\xi) \cdot u_i^1 \tag{7.44}$$

The last term now holds the inactive B-Splines on level 1, which are next to be replaced (removed) with HB-Splines on higher refinement levels. The corresponding coefficients can be recursively determined with the classic subdivision relation. This recursive subdivision procedure is repeated until all coefficients are determined.

7.2.6 Subdivision Projection

7.2.6.1 Main Idea

In finite elements that are constructed through hierarchical refinement, the interpolation within some integration cells, may depend on B-Splines defined across multiple levels. In addition, the number of degrees of freedom per element (intergration cell) is not constant. This occurs specifically on the boundaries, between two consecutive domains of the hierarchy.

In this sense, Bornemann et al have proposed a methodology, in order to efficiently evaluate the element matrices and vectors, by using a constant number of B-Splines in each cell. This number depends on the finest level functions that exist in the area of interest. So, it is possible that regions with different finer level functions will be encountered.

The matrices and vectors are subsequently projected to the correct levels, by using the subdivision relation along with its corresponding weights. Bornemann et al refer to this operation, as **subdivision projection**.

7.2.6.2 Definition of Integration Cells

Before presenting the algorithmic approach, it is necessary to specify accurately how the finest level of functions will be determined. For this purpose, a scalar variable is defined, which will be called level width. To construct this variable, the whole mesh must be scanned, in order to specify the finest level function that has non-zero support over an integration cell. In this sense, level width will be determined as:

$$\Lambda(\xi) = \max_{\ell} \left\{ \ell \geq 0 \,|\, \xi \in \operatorname{supp} N_i^{\ell}, i \in A_c^{\ell} \right\} \tag{7.45}$$

In other words, Λ(ξ) is equal to the local maximum of B-Splines refinement level ℓ, present at the random coordinate ξ. Moreover, Λ(ξ) is a piecewise constant scalar function, due to the nested nature of the subdomains.

Now that Λ has been defined, the integration cells can be also expressed in a more compact and solid way. Thus, integration cells are defined as the union of cells which are on the same level Λ. A formal identification is:

$$IC^{\ell} = \left\{ i \in \mathbb{Z}^d \,|\, \Omega_i^{\ell} \subset \Omega, and : \Lambda\left(\Omega_i^{\ell}\right) = \ell \right\} \tag{7.46}$$

7.2.6.3 Algorithmic Approach

As discussed so far, a B-Spline can be expressed as a linear combination of finer functions through the subdivision relation. From the previous chapter, we also know that by applying several refinement levels, a coarse B-Spline can be expressed by functions of any finer level. Bornemann et al used this observation in order to express all non-zero B-Splines within an integration cell by using functions from one single level, which happens to be the Λ level. As a result, during numerical integration, all element integrals will depend on a fixed number of functions. The subdivision projection weights are subsequently used to map the fixed size element vectors and matrices into the hierarchic approximation space.

So, a formal presentation of the subdivision projection technique is needed, along with the elaboration on the efficient computation of the subdivision projection vectors, which will be extensively described next. Once again, the non-zero functions of the hierarchical basis over an integration cell can be expressed as a linear combination of same level B-Splines N^{Λ}.

$$N_i^{\ell} = N^{\Lambda} s_i^{\ell,\Lambda} \tag{7.47}$$

where $s_i^{\ell,\Lambda}$ is the subdivision vector, associated with the specific function as aforementioned. Also, note that always $\ell \leq \Lambda$ must be valid, in order for the subdivision vectors to be well defined. A computationally convenient approach to determine the projection vector is the subdivision refinement relation. To this end, consider the interpolation of a spline with $N_i^{\ell} \in N_H$ and the finer B-Splines of the finest level Λ, N_i^{Λ}. So:

$$C(\xi) = \sum_i N_i^{\ell}(\xi) \cdot u_i^{\ell} = \sum_j N_j^{\Lambda} \cdot u_j^{\Lambda} = N^{\Lambda} \cdot u^{\Lambda} \tag{7.48}$$

and restricted to the spline segment over the cell $\square_k^{\Lambda}$, only the B-Splines that are non-zero on the cell are needed:

$$u_{\square_k^{\Lambda}} = \sum_{j \in \Im_k^{\Lambda}} \mathrm{N}_j^{\Lambda}(\xi) \cdot \mathrm{u}_j^{\Lambda} = N^{\Lambda} u^{\Lambda} \tag{7.49}$$

The coefficients of the finer level Λ are a linear combination of the coefficients of the coarser level ℓ. When the difference between the levels ℓ and Λ is one, the subdivision projection vector $s_i^{\ell,\Lambda}$ is a row of the previously introduced subdivision matrix. However, when the difference between ℓ and Λ is more than one, $s_i^{\ell,\Lambda}$ represents the result of several steps.

In order to calculate $s_i^{\ell,\Lambda}$, the value 1 is assigned to the control point i (and its corresponding function) at the "coarse" level ℓ and values 0 to all the other same level

control points, which interact with the one that is under evaluation. After one or more levels of refinement, the coefficients $s_i^{\ell,\Lambda}$ can be collected from the locally refined mesh.

7.2.7 Example

Consider an initial coarse knot value vector [0 0 0 1 2 3 4 5 6 7 8 9 10 11 12 12 12] and polynomial degree p=2 as shown in **Fig. 7.7**. In reality, refined cells (knot spans) are identified by an error-estimator along with their corresponding p+1 intersected basis functions. In this case the refined basis functions are chosen carefully to better understand the proposed refinement algorithm. The chosen coarse basis functions are three. Two that interact together (N^1_4, N^1_6) and one that lies in a distance where it is not affected by the refinement of the previous two (N^1_{10}). Each basis function's support knot span is subdivided in two by knot insertion. So, the hierarchical knot value vector after one refinement step will be

[0 0 0 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 7 7.5 8 8.5 9 9.5 10 11 12 12 12]

and the knot vector of level 2 after the necessary knot insertion will be later shown:

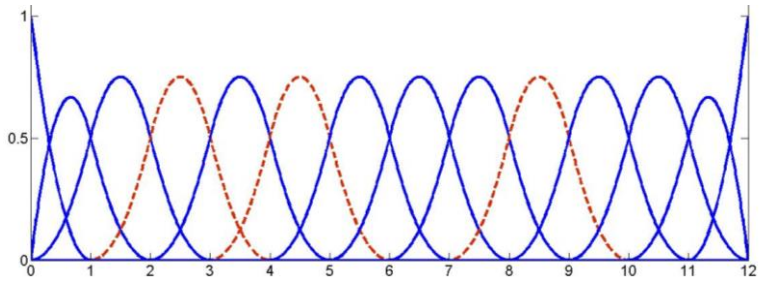


Figure 7.7.
Basis functions with knot value vector [0 0 0 1 2 3 4 5 6 7 8 9 10 11 12 12 12] The red basis functions N^1_4, N^1_6, N^1_{10} will be refined.
(Iakovos Antonios and Karras Dimitrios , 2015)

Every basis function of degree p has (p+2) children whose total support coincides with the maternal function's support. When the refined basis functions are close enough it is possible to share common children basis functions. That is the case of the two first chosen basis functions which will offer the opportunity to present the way of handling such cases. In some other cases some of the refined basis functions' children are able to represent coarse ones as linear combination. The corresponding course basis functions are then redundant and have to be removed to meet the property of linear independence of basis functions. It is obvious that meeting the linear independence property has its cost, which is no other than the disturbance of local nature feature of hierarchical adaptive refinement. When refinement is spread more interactions and degrees of freedom have to be taken into consideration leading to more cost intensive analysis.

The chosen basis functions that will be refined are N_4^1, N_6^1 and N_{10}^1 that belong to the coarsest level 1 and are depicted with red color in **Fig. 7.7**. Basis functions will be identified by their parametric ID.

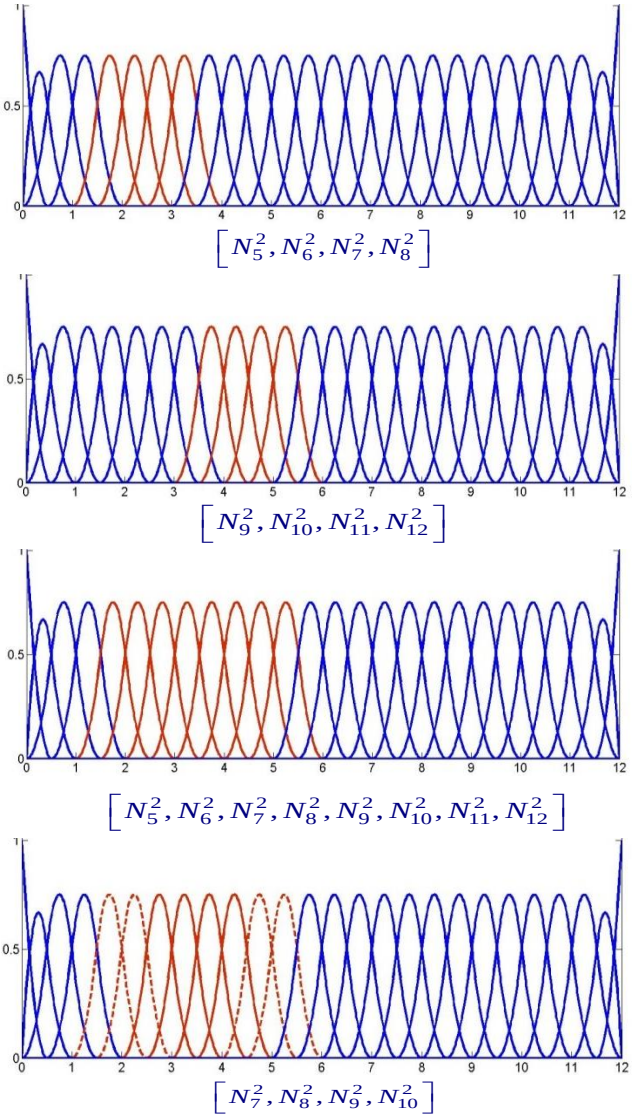


Figure 7.8.

(a), (b) Illustrate the children of N_4^1 and N_6^1 **(c)** The union of the children **(d)** the underlying children $\left[N_7^2, N_8^2, N_9^2, N_{10}^2\right]$ of N_5^1

(Iakovos Antonios and Karras Dimitrios , 2015)

The next basis function to be refined will be chosen from level 2. These basis functions will be replaced by a set of finer ones belonging in level 3. The procedure applied will be the used between levels 1 and 2.

Now it is time to present the active B-Splines indices through the three level. For level 1 (maternal), the active B-Splines are all the existing B-Splines except the ones that have been chosen to be refined. So the active indices are

$$A_c^1 = \left\{ \mathrm{N}_1^1, \mathrm{N}_2^1, \mathrm{N}_3^1, \mathrm{N}_5^1, \mathrm{N}_7^1, \mathrm{N}_8^1, \mathrm{N}_9^1, \mathrm{N}_{11}^1, \mathrm{N}_{12}^1, \mathrm{N}_{13}^1, \mathrm{N}_{14}^1 \right\}.$$

The active indices may alter during compilation step.

The parametric ID of the children is determined as $2i-(p+1),2i-p,...,2i$, where i is the parametric ID of their maternal function. Thus, in the current example these indices will be, $\{5,6,7,8\}$ for N_4^1, $\{9,10,11,12\}$ for N_6^1 and $\{17,18,19,20\}$ for N_{10}^1. These indices can be easily distinguished by simply accompany them by their level number. In this case the children sets are the following:

$$N_4^1 \text{ children:} \left[N_5^2, N_6^2, N_7^2, N_8^2 \right]$$

$$N_6^1 \text{ children:} \left[N_9^2, N_{10}^2, N_{11}^2, N_{12}^2 \right]$$

$$N_{10}^1 \text{ children:} \left[N_{17}^2, N_{18}^2, N_{19}^2, N_{20}^2 \right]$$

Proceeding now to the compilation step, the coarse functions that can be expressed by a linear combination of finer level 2 functions have to be identified and consequently removed. **Fig. 7.8** shows that the union of children of N_4^1 and N_6^1 contains the children of N_5^1 function. As it can be easily noticed the union is:

$$\bigcup children\left(N_4^1, N_6^1\right) = \left\{ N_5^2, N_6^2, N_7^2, N_8^2, N_9^2, N_{10}^2, N_{11}^2, N_{12}^2 \right\} \tag{7.51}$$

where $\left[N_7^2, N_8^2, N_9^2, N_{10}^2 \right]$ are the children of N_5^1.

As a result N_5^1 is redundant and thus will be completely removed. After the removal, the updated active B-Splines matrix will be:

$$A_c^1 = \left\{ \mathrm{N}_1^1, \mathrm{N}_2^1, \mathrm{N}_3^1, \mathrm{N}_7^1, \mathrm{N}_8^1, \mathrm{N}_9^1, \mathrm{N}_{11}^1, \mathrm{N}_{12}^1, \mathrm{N}_{13}^1, \mathrm{N}_{14}^1 \right\} \tag{7.52}$$

Moving forward to the next level of the hierarchy, the active B-Spline indices are:

$$A_c^2 = \left\{ N_5^2, N_6^2, N_7^2, N_8^2, N_9^2, N_{10}^2, N_{11}^2, N_{12}^2, N_{17}^2, N_{18}^2, N_{19}^2, N_{20}^2 \right\} \tag{7.53}$$

We choose to refine N_6^2 and N_9^2 for the next level as shown in **Fig. 7.9.a**. Similar as before, these two functions are being removed immediately from the active functions group. Additionally, (p+2) children are introduced in level 3 for each of the two functions.

The active indices for level 2 so far are:

$$A_c^2 = \left\{ N_5^2, N_{10}^2, N_{11}^2, N_{12}^2, N_{17}^2, N_{18}^2, N_{19}^2, N_{20}^2 \right\}$$

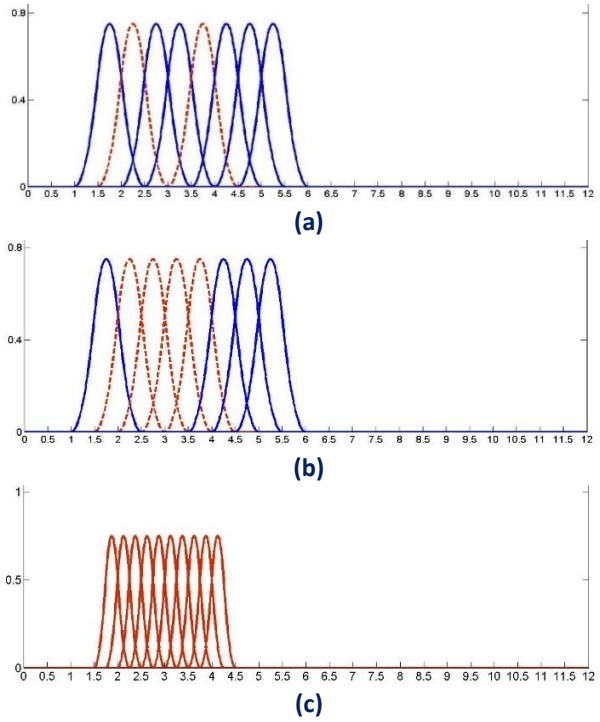


Figure 7.9.
(a) In refinement step 2 N_6^2 and N_9^2 are chosen to be refined **(b)** In compilation step 2 N_7^2 and N_8^2 are also removed as redundant **(c)**The children of level 3created by the subdivision of the selected basis functions of level 2
(b) (Iakovos Antonios and Karras Dimitrios , 2015)

The introduced children of the two refined functions are the following:

$$N_6^2 = \left\{ N_9^3, N_{10}^3, N_{11}^3, N_{12}^3 \right\}$$

$$N_9^2 = \left\{ N_{16}^3, N_{17}^3, N_{18}^3, N_{19}^3 \right\}$$

As can be seen in **Fig. 7.9.b**, the union of the support of the children in level 3 spans across an area. As pointed by Bornemann et al, in this case, when a function of the previous level has its support covered entirely by functions of the next finer level, this function is considered redundant and must be removed. In this example, along with N_6^2 and N_9^2, N_7^2 and N_8^2 will also be removed. Thus, the active B-Spline set for level 2 will finally be:

$$A_c^2 = \left\{ N_5^2, N_{10}^2, N_{11}^2, N_{12}^2, N_{17}^2, N_{18}^2, N_{19}^2, N_{20}^2 \right\}$$

A key issue of the removal of the redundant B-Splines, that were not initially chosen to be refined, is that after their removal, the complete set of their children must be introduced. So, for the removed functions of level 2, the corresponding children will be:

$$N_7^2 \text{ children} : \left\{ N_{11}^3, N_{12}^3, N_{13}^3, N_{14}^3 \right\}$$

and

$$N_8^2 \text{ children} : \left\{ N_{13}^3, N_{14}^3, N_{15}^3, N_{16}^3 \right\}$$

their union will form the active B-Spline set for level 3 shown in **Fig. 7.9.c**, which will be the last one. From the above indices, it can be notice that the new introduced children are N_{13}^3 and N_{14}^3, which are necessary in order to completely remove the redundant coarse functions. So, the last set of active B-Splines for level 3 will be:

$$A_c^3 = \left\{ N_9^3, N_{10}^3, N_{11}^3, N_{12}^3, N_{13}^3, N_{14}^3, N_{15}^3, N_{16}^3, N_{17}^3, N_{18}^3 \right\}$$

Due to the nested nature of hierarchical refinement it is not important to deal with relations between level 1 and 3.

All the above procedures end up in forming the final hierarchical basis. Hierarchical basis is a set of overlaying bases and thus, it is proper to be represented as one. So all the levels will be projected on the same graph in **Fig. 7.10**

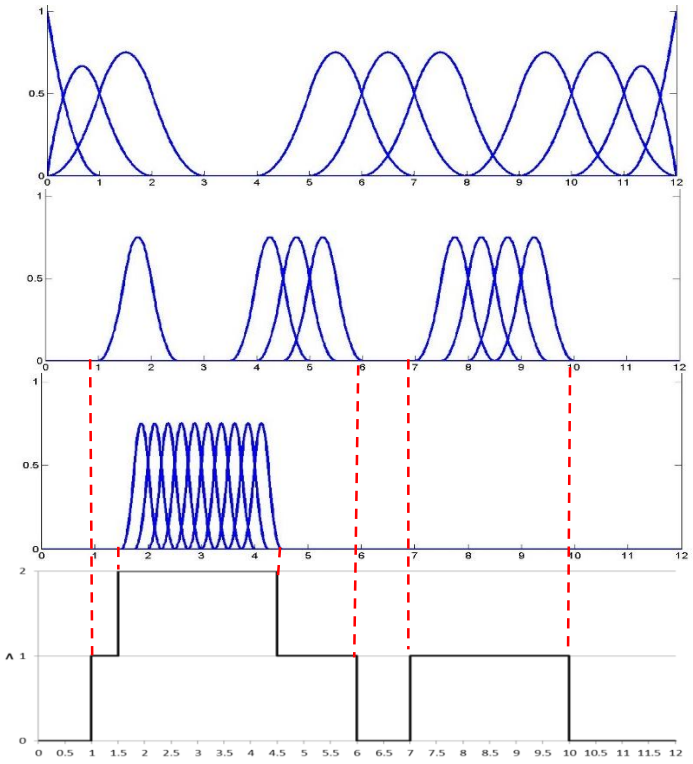


Figure 7.10.
The final hierarchical basis can be seen. In the bottom of the figure the level width Λ is depicted for each integration cell. The number indicate the finest available set of functions in each integration cell.
(Iakovos Antonios and Karras Dimitrios , 2015)

With the hierarchical basis in hand, it is quite easy to notice that the partition of unity is not satisfied along the basis. This phenomenon is usually observed at the boundaries between coarser and finer levels. To tackle this problem of hierarchical bases, scale factors or coefficients have to be introduced to satisfy partition of unity in each integration cell. This can be done by applying the subdivision projection Bornemann proposed.

Each and every one integration cell has to be scanned to find its level width $\Lambda\left(\square_k^{\ell}\right)$, which is equal to the highest level of basis function that intersects the cell. Thus, we can evaluate properly all HB-Splines by expressing the coarse B-Splines of level $\ell < \Lambda$, as linear combination of the finest available set of functions.

In this example, coefficients will be evaluated at the point $\xi = 2.15$. This point belongs to the following knot spans in the three levels of the hierarchy:

- For level 1 in knot span $[2,3]$ where N_3^1 has non-zero value.
- For level 2 in knot span $[2,2.5]$ where N_5^2 has non-zero value.
- In level 3 in knot span $[2,2.25]$ where N_9^3, N_{10}^3 and N_{11}^3 have non-zero values.

It is obvious that the coarse functions of levels 1 and 2 should be expressed by a linear combination of the finest available function set, which happens to be N_9^3, N_{10}^3 and N_{11}^3. However, as the finest functions have not been introduced by coarse functions that are currently active there is need to find a way to relate them properly. By doing so, it is easier to evaluate the coefficients needed for each function.

Any basis function can be written as a linear combination of its children. The children of the basis (for uniform basis) have a certain start and end knot. Thus, by identifying one of those two entities, it is easy to relate the functions of the finest level with the children of the coarse functions. The evaluation will be made in the finest cell that $\xi = 2.15$ belongs to, which is $\square^3 = [2, 2.25)$.

So, the functions are now able to be represented in the specific integration cell as a combination of $\left[N_9^3, N_{10}^3, N_{11}^3 \right]$.

Function N_5^2 can be represented as:

$$N_5^2 = \overbrace{\left[\frac{3}{4}, \frac{1}{4}, 0 \right]}^{s_8^{2,3}} \cdot \left[N_9^3, N_{10}^3, N_{11}^3 \right]$$

Also, by applying the subdivision property relation recursively, we can get the coefficients for N_3^1, which are:

$$\mathrm{N}_3^1 = \underbrace{\left[\frac{5}{8}, \frac{3}{8}, \frac{3}{16} \right]}_{s_3^{1,3}} \left[N_9^3, N_{10}^3, N_{11}^3 \right]$$

The above relation will be explained in a more detailed way, as the recursive procedure may not be as obvious as it seems.

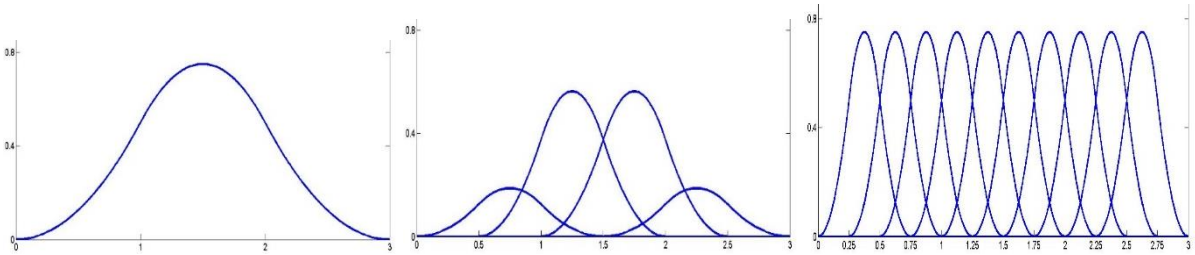


Figure 7.11.
(a) The maternal B-Spline is shown.**(b)** The children for one level of refinement along with their weights.
(c) The children for two level refinement but with unknown weights.
(Iakovos Antonios and Karras Dimitrios , 2015)

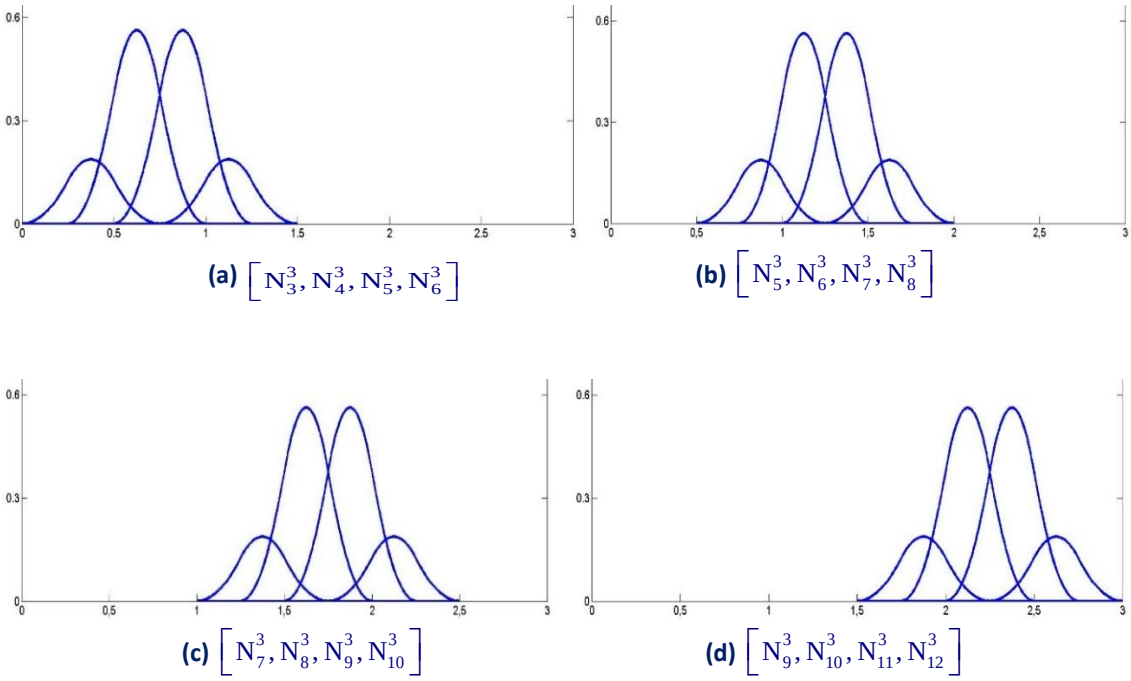


Figure 7.12.
Children of B-Splines in level 3.
The same children may have different weights for different maternal basis functions.
(Iakovos Antonios and Karras Dimitrios , 2015)

As it can be seen from the following table in **Fig. 7.13**, each function is affected by p maternal of the previous level and their corresponding coefficients have to be added. In **Fig. 7.12** are presented the children of level 2 basis functions. It is important to remember that level 2 basis functions are already multiplied with their scale factors. In fact these scale factors multiply the control point coefficients. So it is important to multiply scale factors of level 3 with their corresponding maternal scale factors (level 2) before adding them. The coefficients that accompany the 1st column refer to the relation between level 1-2 (**Fig.**

7.11). The weights that are scattered in the table are referring to the relation between level 2-3 and finally, the coefficients that are written in the last row express the relation of level 3 children with the maternal function N_3^1 in level 1.

	N^3_1	N^3_2	N^3_3	N^3_4	N^3_5	N^3_6	N^3_7	N^3_8	N^3_9	N^3_{10}
0.25	0.25	0.75	0.75	0.25						
0.75			0.25	0.75	0.75	0.25				
0.75					0.25	0.75	0.75	0.25		
0.25							0.25	0.75	0.75	0.25
Final Weights	**0.0625**	**0.1875**	**0.375**	**0.625**	**0.75**	**0.75**	**0.625**	**0.375**	**0.1875**	**0.0625**

Figure 7.13.
The scaled children of N_3^1 after two consecutive refinements.
(Iakovos Antonios, 2015.)

The final weights shown in the previous table have been applied in **Fig. 7.14,** producing the children of B-Spline N_3^1 in level 3.

$$\mathrm{N}_3^1 = \left[\frac{1}{16}, \frac{3}{16}, \frac{3}{8}, \frac{5}{8}, \frac{3}{4}, \frac{3}{4}, \frac{5}{8}, \frac{3}{8}, \frac{3}{16}, \frac{1}{16}\right] \cdot \left[\mathrm{N}_{12}^3, \mathrm{N}_{13}^3, \mathrm{N}_{14}^3, \mathrm{N}_{15}^3, \mathrm{N}_{16}^3, \mathrm{N}_{17}^3, \mathrm{N}_{18}^3, \mathrm{N}_{19}^3, \mathrm{N}_{20}^3, \mathrm{N}_{21}^3\right]^T$$

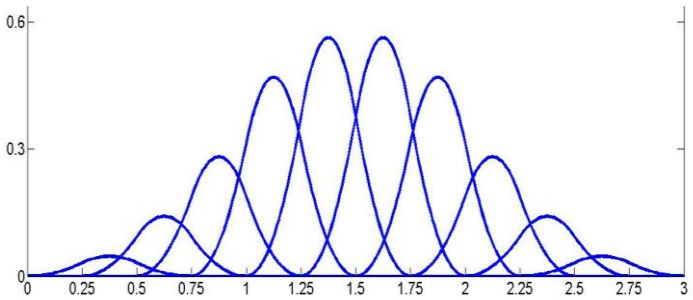


Figure 7.14.
The scaled children of N_3^1 after two consecutive refinements.
(Iakovos Antonios and Karras Dimitrios , 2015)

Finally, the more obvious evaluations, which refer to $N_9^3, N_{10}^3, N_{11}^3$, is done as follows:

$$N_{18}^3 = \underbrace{[1,0,0]}_{s_{18}^{3,3}}\left[N_9^3, N_{10}^3, N_{11}^3\right]$$

$$N_{10}^3 = \underbrace{[0,1,0]}_{s_{19}^{3,3}}\left[N_9^3, N_{10}^3, N_{11}^3\right]$$

$$N_{11}^3 = \underbrace{[0,0,1]}_{s_{20}^{3,3}}\left[N_9^3, N_{10}^3, N_{11}^3\right]$$

The evaluation in each cell is different for each coarse function. Thus, the same procedure must be done several times for the B-Splines of the coarser levels $\ell < \Lambda\left(\square_k\right)$. In addition to that, the existing children in each level have different weights for each coarse function. This does not allow us to keep the same weights for more than one function and consequently the evaluations cannot be skipped in any way.

Having evaluated the coefficients for cell $[2, 2.25]$, the weights can be addressed e weights to the control points and form the stiffness matrix. So, in terms of interpolation of a spline curve $C(\xi)$, the subdivision projection has the following consequences. The spline segment over the integration cell $\square = [2, 2.25)$ uses the following coefficients:

$$C(\xi)_{\square=[2,2.25)} = N_3^1 \cdot u_3^1 + N_5^2 \cdot u_5^2 + N_9^3 \cdot u_9^3 + N_{10}^3 \cdot u_{10}^3 + N_{11}^3 \cdot u_{11}^3$$

(7.50) can be equally written as following by using the subdivision projection vectors found previously:

$$C(\xi)_{\square=[2,2.25)} = \left[N_9^3, N_{10}^3, N_{11}^3 \right] \left[s_3^{1,3} u_3^1 + s_5^{2,3} u_5^2 + s_9^{3,3} u_9^3 + s_{10}^{3,3} u_{10}^3 + s_{11}^{3,3} u_{11}^3 \right] \tag{7.54}$$

The element matrices and vectors are multiplied with subdivision weights during the assembly stage of the global matrices and vectors. As a result, subdivision projection allows the reuse of conventional finite element implementations by only using B-Splines at a fixed level as basis functions.

7.2.8 Conclusions

In conclusion, HB-Spline refinement proceeds by successfully replacing selected functions, with finer B-Splines that ensure geometry remains the same by following the explained three step procedure. HB-Spline ensure another very important property which is the linear independence of basis functions and consequently degrees of freedom. Linear independence is a key aspect of analysis. The easy implementation of HB-Splines is derived from the use of subdivision property which connects basis functions from different hierarchical levels. All the above features of HB-Splines result in sufficient and accurate solutions with significantly less degrees of freedom than uniform refinement. An also attractive property of HB-Splines is the lossless transfer of field variables during the incremental/ iterative solution procedure for non-linear problems.

7.3 TRUNCATED B-SPLINES AS PROPOSED BY GIANNELLI

7.3.1 Introduction

The technique of hierarchical truncated B-Splines (THB-Splines) has been proposed by Giannelli in 2012. The aim was to suitably modify the classical hierarchical B-Splines basis to define basis functions with reduced local support that satisfy partition of unity property. Truncation decreases the overlapping of basis function's supports which arise from the enrichment of the basis with the introduction of finer basis functions. In order to completely understand the proposed technique, proofs and examples will be given, as well as some numerical results.

7.3.2 Truncation Technique

Giannelli et al, have based their proposal in the existing knowledge presented by Vuong. The construction of hierarchical B-Splines can be modified by suitably truncating basis functions according to finer levels in the hierarchy. It is known that classical hierarchical refinement schemes have some negative features. One of them is the formation and solution of linear systems associated to the discrete variational problem, as a result of the extended overlapping nature of basis functions. Higher number of overlaps suggests denser system matrices which lead to the increase of needed computational cost.

What really differs truncated hierarchical B-Splines from the classic hierarchical procedures is that after representing each coarse basis function as linear combination of finer ones follows the elimination from this representation of the contribution corresponding to the subset of finer basis functions that are effectively included in the hierarchical basis. All these lead to the truncation mechanism preserving all the good properties of hierarchical B-Splines, such as linear independence and non-negativity. In addition, THB-Splines:

- Have smaller support
- Form a partition of unity by their construction, so no weights are needed to maintain it.

In addition to the aforementioned characteristics, another direct result of the truncated B-Splines is the reduced sparseness in stiffness and mass matrices due to the decrease of basis functions' overlapping. In Giannelli's proposal some of Kraft's restrictions are skipped. An example of this fact is that in truncation, refinement is allowed close to the subdomain boundaries.

The truncation mechanism works as following:

Let $N_i^\ell \in N^\ell$ a function defined at level ℓ and also let:

$$N_i^\ell = \sum_{j:N_j^{\ell+1} \in N^{\ell+1}} a_j \cdot N_j^{\ell+1} \tag{7.55}$$

be its representation respect to the fine scale basis associated to level ℓ+1. The truncation of N_i^{ℓ} with respect to $N^{\ell+1}$ is defined as:

$$trunc^{\ell+1}\left(N_i^{\ell}\right) = \sum_{\substack{j:N_j^{\ell+1}\in N^{\ell+1} \\ \text{suppN}_j \not\subset \Omega^{\ell+1}}} a_j \cdot N_j^{\ell+1}\ell \tag{7.56}$$

From the above relations is clear that the coefficients a_j depend not only on the component $N_j^{\ell+1}$ they refer to, but also on the function N_i^{ℓ}. The truncation as given in (7.56) is called additive representation. It is also possible to use a subtractive representation, expressing the truncation as:

$$trunc^{\ell+1}\left(N_i^{\ell}\right) = N_i^{\ell} - \sum_{\substack{j:N_j^{\ell+1}\in N^{\ell+1} \\ \text{suppN}_j \subseteq \Omega^{\ell+1}}} a_j \cdot N_j^{\ell+1} \tag{7.57}$$

Both representations lead to the exact same hierarchical truncated basis, but they also have different advantages and disadvantages, which refer to the procedures of identification and evaluation. One important observation is that for the 2D or 3D case, the truncation mechanism may lead to non-rectangular support. The behavior of this property is a subject still under investigation.

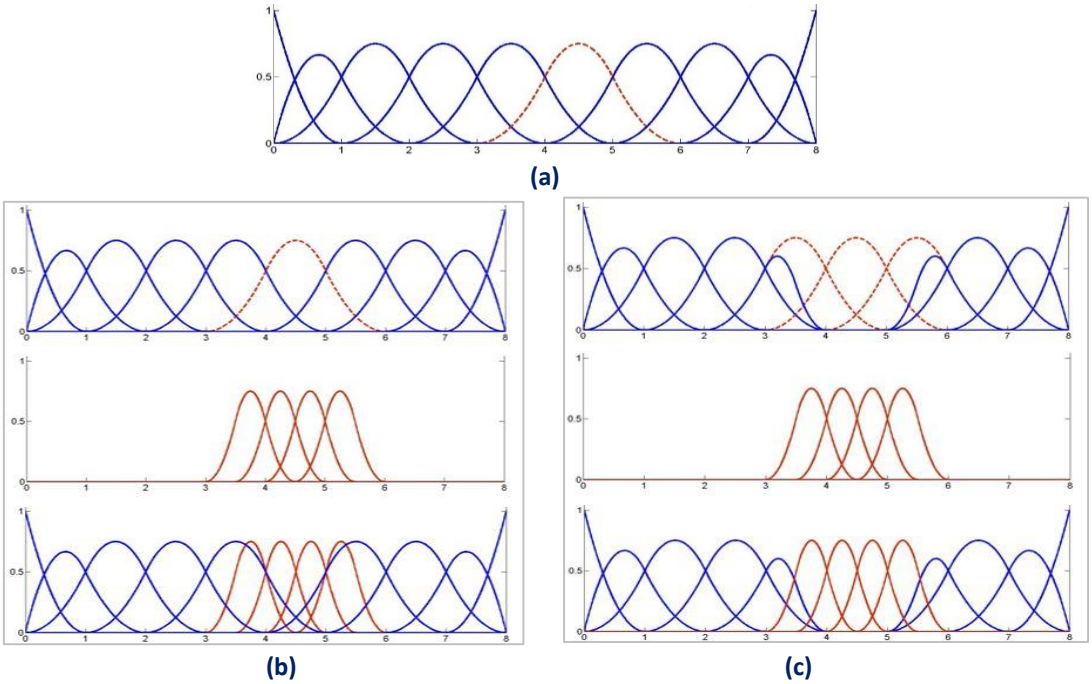


Figure 7.15.
(a) Chosen basis function for refinement.**(b)** Classical hierarchical basis construction **(c)** Truncated Hierarchical Basis Construction
(Iakovos Antonios 2015)

So by utilizing the above mechanisms, the truncated hierarchical basis can be recursively constructed by the following three basic steps.

I. Initialization:

$T^0 = H^0$, where H refers to the classical hierarchical basis

II. Recursive Case:

$T^{\ell+1} = T_A^{\ell+1} \cup T_B^{\ell+1}$ for $\ell = 1,\ldots,L-1$ where :

- $T_A^{\ell+1} = \left\{ trunc^{\ell+1}\left(T_i^{\ell}\right) : T_i^{\ell} \in T^{\ell} \wedge \operatorname{supp}\left(T_i^{\ell}\right) \not\subset \Omega^{\ell+1} \right\}$
- $T_B^{\ell+1} = H_B^{\ell+1}$

III. $T = H^L$

In step **(I)**, we copy the hierarchical basis of level 0 (maternal) to the truncated hierarchical basis.

In the recursive case, coarse basis functions, whose support has a non-empty overlap with Ω^{l+1}, are truncated. Actually, truncation mechanism is activated usually for those functions, which lie on the boundaries between two subdomains. Another case, where truncation may be activated, is when random finer B-Splines are introduced, which cannot describe completely a coarse B-Spline and thus remove it. The other half of this step is the same as in the classical hierarchical refinement algorithm. The anchors for the classical and truncated hierarchical basis functions will coincide.

Now that the basic algorithmic procedure has been presented, some extra comments about the use of either truncation mechanisms will be made. It has been noticed that programming the two methods yields important differences and consequences.

In a typical finite element code, one has to deal with two important aspects:

- determining which basis functions are active over a given element (cell)
- evaluating such functions at the boundaries between two subdomains.

To address the former aspects, it is essential to find a convenient way to retrieve or store the support of the functions. In general, the data organizing in hierarchical refinement seems to be as important as the chosen implementation methodology. In the case of B-Splines, this is generally easy since the support is identified by the local knot vectors. When the B-Spline is a standard tensor product and the support is therefore rectangular, this becomes even easier, since one only needs to check the starting and ending knots of the local knot vectors, as we aforementioned. However, THB-Splines do not always have rectangular support making the data organizing more complex than usual.

The subtractive representation (7.57) is unfortunately not very helpful in this sense. This happens because when a given component is subtracted, it is not automatically guaranteed that the function itself is vanished in that area.

Given an element $\square_k^\ell \in \mathrm{supp}\left(N_i^\ell\right)$, we should check if all possible components on that element are removed in order to know if $trunc\left(N_i^\ell\right)$ has support on $\square_k^\ell$ or not. So, the subtractive representation does not allow easy identification of the function's support.

On the other hand it seems it is more efficient in the evaluation of basis functions. Evaluating basis functions in an efficient and fast way seems to be a very important issue in the isogeometric setting. Since Cox-de-Boor algorithm is a typical bottleneck of the code, we would generally like to perform as few function evaluations as possible.

In this case, subtractive representation is not efficient. In a biquadratic case, a representation in terms of next-level basis functions comprises $(2+2)^2=16$ children. This number obviously increases along with the increase of the polynomial degree: 25 for bicubic functions, 36 for biquartic and so on. Moreover, an additive representation may acquire to store the function in terms of the finest available scale, which would increase the amount of needed components.

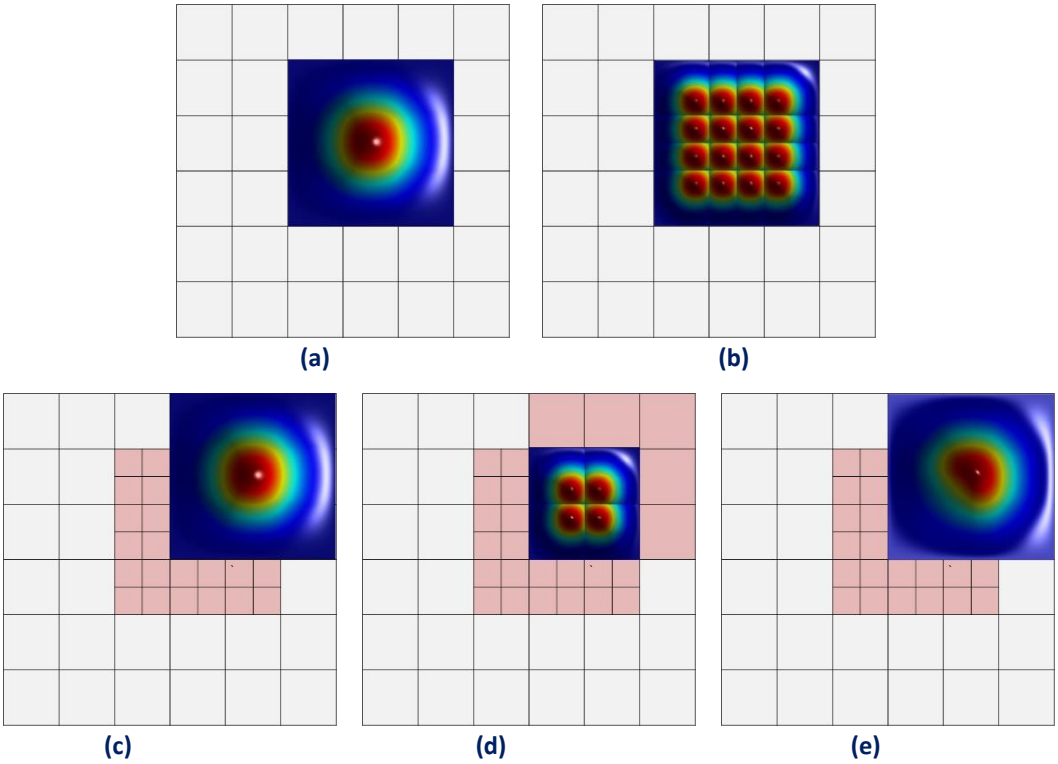


Figure 7.16.

(a) Maternal basis function **(b)** Corresponding subsidiary fine basis functions **(c)** Coarse basis function. Part of its support is common with the support of the introduced subsidiaries **(d)** Common subsidiaries that have to be subtracted from the upper right basis function **(e)** Truncated basis functions after the subtraction of four basis functions presented in (d)

To give an example, let us recall a biquadratic function. For each of the truncated hierarchical basis functions only 4 components are removed as in **Fig. 7.16**.

This consequently means that in an additive representation 21 fine-scale functions had to be evaluated in order to compute the value of the chosen B-Spline. In contrast, in a subtractive representation only five functions would have been evaluated: The original tensor product B-Spline along with the four children that would be subtracted. To summarize it is obvious that the additive representation is useful, when determining the support but inefficient in the function evaluation process.

From the above recursive algorithm derive the following properties:

- The cardinality of the basis is the same for the classical hierarchical and the truncated basis.
- The spanned spaces are the same.
- The basis functions remain non-negative.
- For each truncated basis function T_i^ℓ introduced at level ℓ, there exists one B-Spline $N_j^\ell \in N^\ell$ which satisfies:

$$Tr_i^\ell = trunc^{L-1}\left(trunc^{L-2}...\left(trunc^{\ell+1}\left(N_j^\ell\right)\right)...\right) \tag{7.58}$$

- The truncated basis functions are linearly independent and form a partition of unity.

Proof: The functions of the truncated basis are linearly independent.

It has to be proven that:

$$\sum_{Tr_j^\ell \in T^\ell} a_j Tr_j^\ell = 0 \Rightarrow a_j = 0$$

The sum on the left can be decomposed, according to the maternal B-Splines introduced at a certain level ℓ:

$$\sum_{Tr_j^\ell \in T^\ell,\text{mother}\left(Tr_j^\ell\right)\in N^0} a_j Tr_j^\ell + \sum_{Tr_j^\ell \in T^\ell,\text{mother}\left(Tr_j^\ell\right)\in N^1} a_j Tr_j^\ell + ... + \sum_{Tr_j^\ell \in T^\ell,\text{mother}\left(Tr_j^\ell\right)\in N^{l-1}} a_j Tr_j^\ell = 0 \tag{7.59}$$

The basis functions collected by the first sum are the only non-zero functions acting on the region given by Ω^0/Ω^1. In virtue of the previous properties and of the local linear independence of the B-Spline basis, these functions are locally linearly independent on Ω^0/Ω^1 and thus the corresponding coefficients must be zero.

Excluding the functions already considered in this first sum, the basis functions collected by the second sum are the only non-zero functions which act on Ω^1/Ω^2. As before, the corresponding coefficients must be also zero. We can repeat the same argument of sums related to the basis functions whose maternal B-Splines belong to $N^\ell, \ell = 2,...,L-2$.

On the other hand, the basis functions $Tr_i^\ell \in \mathrm{T} \cup \mathrm{N}^{L-1}$ represent just the subset of B-Splines in N^{L-1}, whose support is completely contained in Ω^{L-1}. Hence, they are locally linearly independent on this last subdomain and the coefficients in the last sum must be also zero.

Proof: The truncated hierarchical B-Spline basis T forms partition of unity.

It has to be proven that:

$$\sum_{Tr_i^{\ell} \in T^{\ell}} Tr_i^{\ell} = 1 \text{ on } \Omega^0 \Rightarrow \sum_{Tr_i^{\ell+1} \in T^{\ell+!}} Tr_i^{\ell+1} = 1 \text{ on } \Omega^0$$

It is known that:

$$\sum_{N_i^{\ell} \in N^{\ell}} N_i^{\ell} = 1 \text{ on } \Omega^0 \text{ with } \ell = 0,1,...,L\text{-}1$$

The partition of unity can be shown by induction on the hierarchical level ℓ. The base case simply follows form the above relation with ℓ = 0.

The inductive step:

$$\sum_{Tr_i^{\ell} \in T^{\ell}} Tr_i^{\ell} = 1 \text{ on } \Omega^0 \Rightarrow \sum_{Tr_i^{\ell+1} \in T^{\ell+!}} Tr_i^{\ell+1} = 1 \text{ on } \Omega^0 \tag{7.60}$$

can be proved by re-arranging the sums as follows:

$$\sum_{Tr_i^{\ell} \in T^{\ell}} Tr_i^{\ell} = 1 = \sum_{Tr_i^{\ell} \in T^{\ell}} \sum_{N_j^{\ell+1} \in N^{\ell+1}} a_{N_j^{\ell+1}}^{\ell+1}\left(Tr_i^{\ell}\right) \cdot N_j^{\ell+1} \tag{7.61}$$

(7.63) can be further expanded as:

$$\sum_{Tr_i^{\ell} \in T^{\ell}} Tr_i^{\ell} =$$

$$= \sum_{Tr_i^{\ell} \in T^{\ell}} \left(\sum_{N_j^{\ell+1} \in N^{\ell+1}, \operatorname{supp} N_j^{\ell+1} \not\subset \Omega^{\ell+1}} a_{N_j^{\ell+1}}^{\ell+1} \cdot \left(Tr_i^{\ell}\right) \cdot N_j^{\ell+1} + \sum_{N_j^{\ell+1} \in N^{\ell+1}, \operatorname{supp} N_j^{\ell+1} \subseteq \Omega^{\ell+1}} a_{N_j^{\ell+1}}^{\ell+1} \cdot \left(Tr_i^{\ell}\right) \cdot N_j^{\ell+1} \right)$$

$$= \sum_{Tr_i^{\ell} \in T^{\ell}} \left(\sum_{N_j^{\ell+1} \in N^{\ell+1}, \operatorname{supp} N_j^{\ell+1} \not\subset \Omega^{\ell+1}} a_{N_j^{\ell+1}}^{\ell+1} \cdot \left(Tr_i^{\ell}\right) \cdot N_j^{\ell+1} \right) +$$

$$+ \sum_{N_j^{\ell+1} \in N^{\ell+1}, \operatorname{supp} N_j^{\ell+1} \subseteq \Omega^{\ell+1}} \left(\sum_{Tr_i^{\ell} \in T^{\ell}} a_{N_j^{\ell+1}}^{\ell+1} \cdot \left(Tr_i^{\ell}\right) \right) \cdot N_j^{\ell+1}$$

In the first sum of the last equation, the term in brackets is just $trunc^{\ell+1}\left(Tr_i^{\ell}\right)$. Moreover by swapping the order of sums in the first line of the above relation:

$$\sum_{N_j^{\ell+1} \in N^{\ell+1}} N_j^{\ell+1} = 1$$

$$\sum_{N_j^{\ell+1}\in N^{\ell+1}}\left(\sum_{Tr_i^{\ell}\in T^{\ell}} a_{N_j^{\ell+1}}^{\ell+1}\cdot\left(Tr_i^{\ell}\right)\right)\cdot N_j^{\ell+1}=1 \tag{7.62}$$

So, by comparing the coefficients and by the linear independence of the B-Splines:

$$\sum_{Tr_i^{\ell}\in T^{\ell}} a_{N_j^{\ell+1}}^{\ell+1}\cdot\left(Tr_i^{\ell}\right)=1 \tag{7.63}$$

for all $N_j^{\ell+1}\in N^{\ell+1}$ and in particular for each $N_j^{\ell+1}$ such that $\operatorname{supp} N_j^{\ell+1}\subseteq\Omega^{\ell+1}$. Finally we obtain:

$$1 = \sum_{Tr_i^{\ell}\in T^{\ell}} trunc^{\ell+1}\left(Tr_i^{\ell}\right)+\sum_{N_j^{\ell+1}\in N^{\ell+1},\operatorname{supp} N_j^{\ell+1}\subseteq\Omega^{\ell+1}} N_j^{\ell+1}= \tag{7.64}$$

$$=\sum_{Tr_i^{\ell+1}\in T_A^{\ell+1}} Tr_i^{\ell+1}+\sum_{Tr_i^{\ell+1}\in T_B^{\ell+1}} Tr_i^{\ell+1}=\sum_{Tr_i^{\ell+1}\in T^{\ell+1}} Tr_i^{\ell+1} \tag{7.65}$$

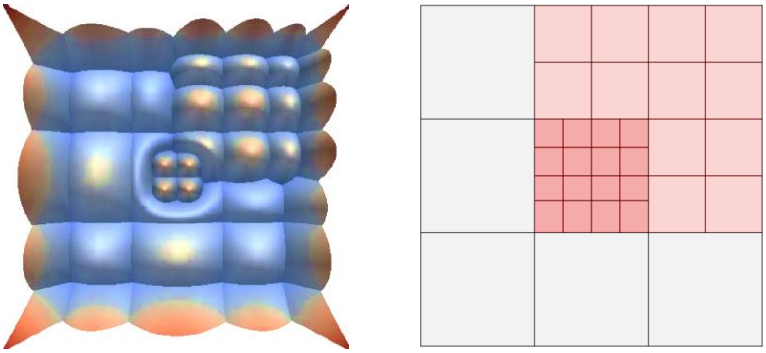
In view of the non-negativity of truncated basis functions and of the previous theorem, the functions in T form a convex partition of unity.

7.3.3 Evaluating Truncation Weights

Yet again, Giannelli's proposal considers some weights used in both additive and subtractive representation of the truncation mechanism. As Vuong proposed, these coefficients are not determined by some mechanism. One idea for evaluating these coefficients is to adopt the subdivision property relation along with the subdivision projection technique. When this task has been performed, active finer B-Splines can be subtracted from the coarse ones in order to create the THB-Splines of each level.

In this sense:

- Classical hierarchical refinement is performed initially
  The entire procedure remains the same until the evaluation step.
- After that, the weights found by the evaluation are used in order to multiply and then subtract the existing active B-Splines of the finest available level, from the coarser active functions.
- Having secured that the redundant B-Splines are completely removed from each level in order to secure linear independence, the subtraction will take place for most of the cases in the boundaries between Ω^{ℓ} and $\Omega^{\ell+1}$ with $\ell<\Lambda$.



(a) (b)

Figure 7.17.
(a) Illustration of a complete truncated hierarchical basis, for two consecutive levels of refinement.
(b) Corresponding element mesh. Finer level elements are depicted with darker shade.
(Giannelli et al, 2014)

7.3.4 Brief Comparison with HB-Splines

In this section, numerical results that have been occurred from the implementation of truncated hierarchical basis are gathered and compared. In general, these results concern the behavior of stiffness and mass matrices. For example, their condition number, their sparsity or even their Cholesky factorizations are some features to compare. Quality of produced meshes along with their hierarchical basis functions can be also examined.

7.3.4.1 Basis functions

For the 1D case, truncation seems to be an ideal technique in order to overcome the drawbacks of classical hierarchical refinement. More specifically, no weights are needed anymore in order for the basis to form partition of unity in each knot span or generally in each integration cell. However, from what the plotting shows, THB-Splines are non-uniform in general as they span across knot spans of various lengths. The knot spans depending on their length belong to different levels of the hierarchy and consequently to different nested domains. This may cause some issues in saving and handling the THB-Splines as their start and end knots will not be known before the truncation.

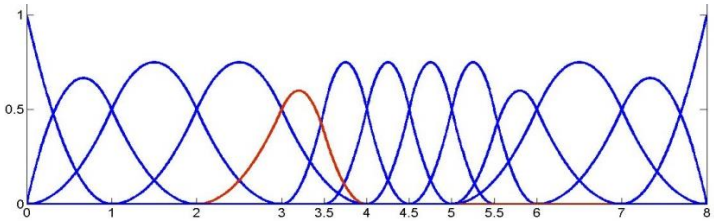


Figure 7.18.
The pointed THB-Spline of degree p=2 has the non-uniform support [2 , 3, 3.5 , 4] and it forms a partition of unity with the all its overlapping functions.
(Iakovos Antonios. 2015)

In 2D and 3D cases, as it is already known, tensor product B-Splines and HB-Splines have rectangular projections of their support in the planes created by two of the parametric axes.

However, in truncation, this is not the case as THB-Splines belong to more than one subdomain and tend to lose their rectangular support due to their truncation.

7.3.4.2 Stiffness matrix

In classical hierarchical refinement every control point can be replaced with $(p+2)^d$ finer control points, where p is the polynomial degree and d the number of problem's dimension. It is straightforward that, as polynomial degree and number of dimensions increase, the removal of one control point equals with the introduction of an increasing number of new control points. New relations between the coarse and fine control points must be taken into consideration in the formulation of global stiffness matrix.

As it was aforementioned in this chapter, truncation enables functions to reduce their support as well as to adapt their values in order to satisfy partition of unity across the basis. The first feature seems to be critical for the comparison between the classical hierarchical and truncated stiffness matrices. Due to their reduced support, the overlapping between truncated and finer functions is smaller. As a result stiffness non-zero values are less. Sparsity and matrix bandwidth are also reduced as shown in **Fig. 7.19**. All the aforementioned aspects can lead to reduction of the needed computational cost for the solution of problems' equations.

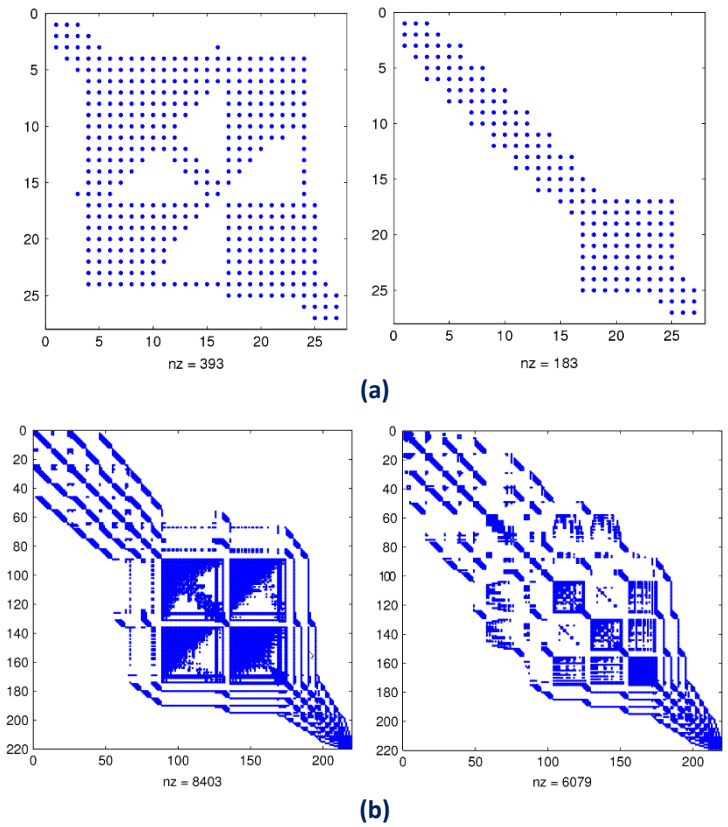


Figure 7.19.
(a) 1D stiffness matrices for classical hierarchical and truncated hierarchical refinement.
(b) 2D stiffness matrices for classical hierarchical and truncated hierarchical refinement.
(K.A. Johannessen et al. 2015)

7.4 Local Hierarchical refinement proposed by Schillinger et al

7.4.1 Introduction

In continuation to the methods described previously, Schillinger et al proposed an alternative local refinement technique which is based on the concepts of HB-Splines, subdivision property and classical FEA knowledge and aims to offer a new straightforward algorithmic approach for adaptive refinement in isogeometric analysis.

The new aspect Schillinger introduced, is the overlaying of finer basis functions with a systematic way that does not always lead to the removal of coarser ones. In his proposal subdivision property is used to connect different hierarchical level basis functions. As it will be later discussed, a more local refinement can be achieved in comparison to classical hierarchical or truncated hierarchical methods. Schillinger also suggests that the refinement should be made to the entity of an element and not of a B-Spline function as in previous methodologies.

7.4.2 Local refinement methodology

7.4.2.1 1D finer functions patch overlay

The refinement algorithms explained until now, were based on the following assumption. The error estimator points out the elements (knot span, area, volume) with high error, via norms or energy theorems. As soon as these elements are identified, the B-Spline functions, with non-zero values over these cells, are chosen for refinement. These functions are completely removed from the coarse basis and their children are introduced to the next finer levels to satisfy the linear independence of basis functions. The only parameter that remains to be addressed is partition of unity. This procedure was named by D. Schillinger subdivision refinement, as it utilizes completely the subdivision property of B-Splines in order to easily satisfy linear independence.

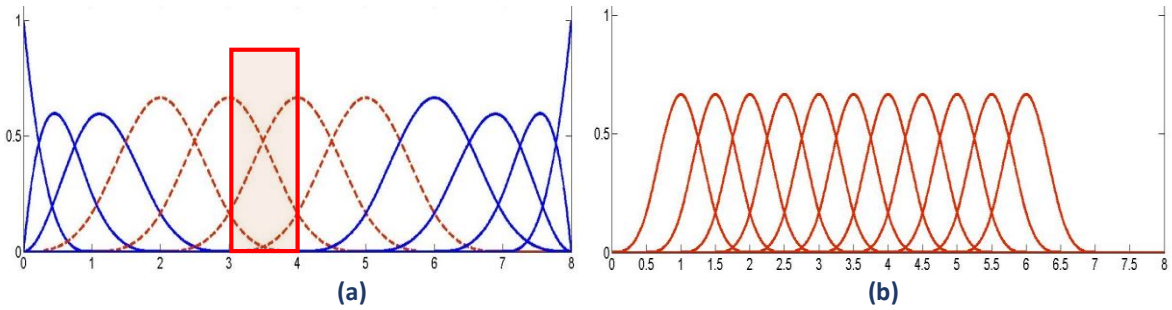


Figure 7.20.

Subdivision refinement. In **(a)** the corresponding refined B-Splines for the selected element (shaded area) are depicted with dashed lines. In **(b)** the introduced children from the subdivision relation are illustrated with red (Iakovos Antonios and Karras Dimitrios , 2015)

Of course, the easy handling of the basis with subdivision refinement comes along with a cost. More specifically, by refining all the overlapping functions of an element the desired local nature is diminished as shown in **Fig. 7.20**. This happens, because a function in IGA overlaps $(p+1)^d$ elements. As the polynomial degree of functions and dimensions of the problem increase, the area that is collaterally refined increases along with them. Consequently, more B-Spline evaluations must be made for both classical and truncated hierarchical refinement in this sense. As it will be presented, Schillinger proposed an efficient way in order to deal with this side effect of the common refinement algorithms.

The author defines a nucleus operation, from which the development starts the refinement of one knot span element. As it is known so far, when refining a function, B-Spline children with contracted support cover the whole support of the refined coarse function. The children are being superposed by adding their scale factors from subdivision property. This leads to the complete hierarchical basis in the center of the refinement region, while at the boundaries the scale factors are used in order to satisfy partition of unity. However, the direct subdivision refinement strategy results in large spread of the refinement beyond the bounds of the knot span element that was initially chosen for refinement. This side-effect obstructs with the local nature we want to achieve.

For this reason, D. Schillinger et al instead of introducing finer children that would emerge from subdivision property, they added an overlay of a fixed number of p^d B-Splines as shown in **Fig. 7.21**. These finer B-Splines would have reduced knot span width and thus support in comparison with the functions of the coarse level (the relations for the knot spans can be found in the previous chapter). Up to this point, no changes to the coarse basis functions are needed, since we know that for a function to be removed $(p+2)^d$ children must exist in order to describe it completely. The hierarchical basis is constructed from the full set of children and all of the coarse functions. The amplitude of the contracted B-Splines does not need to change, thus the presence of scaling factors is ignored.

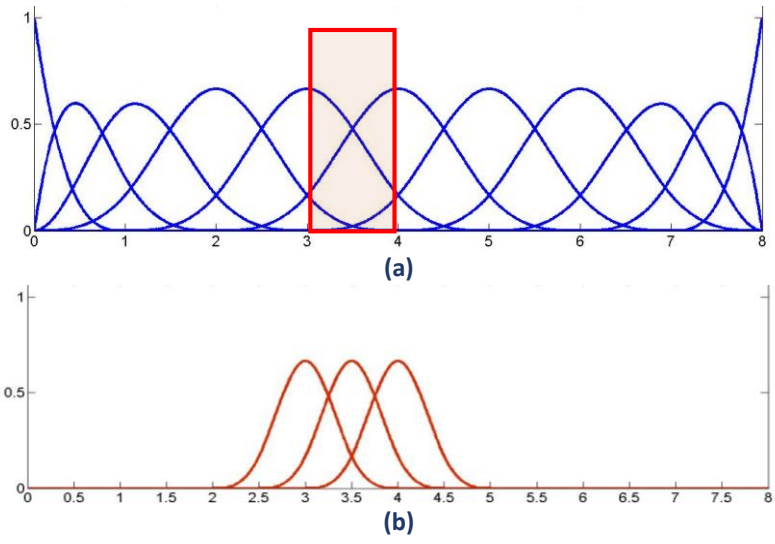


Figure 7.21.

(a) The selected element is shown in grey, but without refining any coarse level B-Spline.**(b)** The overlaying patch is illustrated, along with the p contracted B-Splines
(Iakovos Antonios and Karras Dimitrios , 2015)

Let's now assume that two consecutive knot span elements are going to be refined. Recalling the previous paragraph, one patch of finer contracted B-Splines is introduced for each knot span respectively. As one can easily notice, some of the functions of the two set of B-Splines may be exact the same. This obviously means that for the specific level, there is no need to add the same contracted B-Spline for a second time. Having done this, now the finer functions that have been introduced are (p+2). Consequently, we have just created a full set of contracted B-Splines, which are able to represent at least one coarse function. Thus, the coarse function is redundant and has to be removed in order to restore the linear independence of the hierarchical basis as shown in **Fig. 7.22**.

In particular, this procedure does not affect the higher-order continuity of the refined basis, since the first p-1 derivatives of the hierarchical B-Spline basis functions are zero at their support boundaries. The locality of this method can be accomplished by p children for each element for each overlay patch. More children would spread the refinement like the subdivision refinement. Less children would not be able to remove any coarse function.

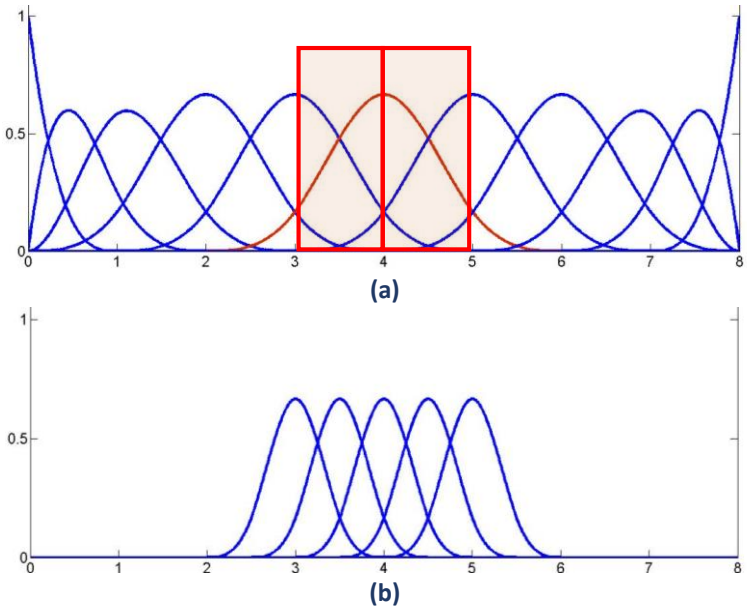


Figure 7.22.
(a) The two refined elements are repsresented by the shaded area. The coarse level B-Spline which will be removed as redundant when the overlaying patch is introduced is depicted with red color. **(b)** the overlaying patch for the first level of refinement
(Iakovos Antonios and Karras Dimitrios , 2015)

7.4.2.2 Removing Multiplicities

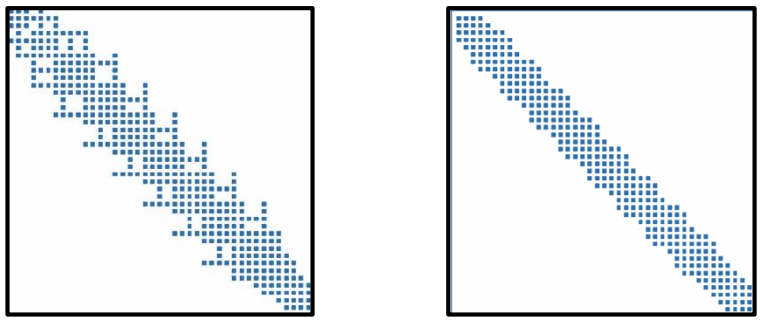
As in classical hierarchical refinement, an operation is necessary in order to restore partition of unity at the boundaries between the two overlaying patches. In the aforementioned refinement proposals this could be achieved with corresponding weights for each subsidiary B-Spline basis function. Schillinger uses a mechanism similar to truncation.

To do this, one must just remove the multiplicities over one element. Starting from $\ell = 1$ (maternal level), it's easy to proceed with the following algorithmic procedure. More specifically, this can be achieved by:

- Checking each hierarchical B-Spline of the next level $\ell + 1$ to determine if it has a common support with a B-Spline of the current level ℓ.

- If it does, it should be checked whether the function of $\ell + 1$ is part of linear combination of subdivision B-Splines that result from the classical subdivision property relation. It is possible that even though one function may have an overlap with a coarse function, it may not be a child that is derived from the subdivision property relation.

- If this happens to be also true and the contracted function under examination can be obtained by the subdivision algorithm, the finer function is multiplied by its scaling factor (subdivision weight) from the B-Spline of the coarser level and subtracted from the coarse one.



(a) **(b)**

Figure 7.23.
Stiffness matrices for classical hierarchical refinement over the diagonal of a regular parametric space. **(a)** Stiffness matrix before the subtraction process. **(b)** Stiffness matrix after the subtraction. (Schillinger et al, 2012)

The subtraction procedure of removing the multiplicities has a beneficial effect on the bandwidth, sparsity and the condition number of the stiffness matrix. This is generally happening because by subtracting certain parts from the coarser levels, the overlaps between B-Splines are reduced. Consequently the interactions between control points are also reduced. The non-zero entries are decreasing as well. A representative example of stiffness matrices is depicted in **Fig. 7.23**.

Finally, to exclude any effect from weak boundary conditions, uniform boundary basis functions are replaced by interpolatory basis functions created from **open** knot vectors, so that *Dirichlet* constraints can be strongly satisfied.

Multi-level hierarchical refinement can be also achieved by recursively applying the previous procedure. More specifically, in each refinement step, the nucleus operation is applied to elements of the currently finest level ℓ, in order to produce a new overlay patch

in level $\ell+1$. The resulting grid consists of a nested sequence of bisected knot span elements and multiple hierarchical overlay levels of repeatedly contracted uniform B-Splines. As aforementioned, the selection of elements is the result of an error-estimator.

7.4.2.3 1D Case Example

In this unit, an extensive 1D example based on Schillinger's methodology will be presented. In contrast to all previous methods Schillinger identifies elements and not basis functions to be refined. As aforementioned, the subdivision property is not fully used to create a next level overlay patch of finer p+2 (the least) contracted functions. More specifically, a patch with p+2 necessary functions is created only when a set of neighbor refined elements is chosen to maintain the local nature of hierarchical refinement.

For this example consider a basis with functions of polynomial degree=3 obeying to the following knot value vector:

$$\Xi = \{0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 11 \quad 11 \quad 11\}$$

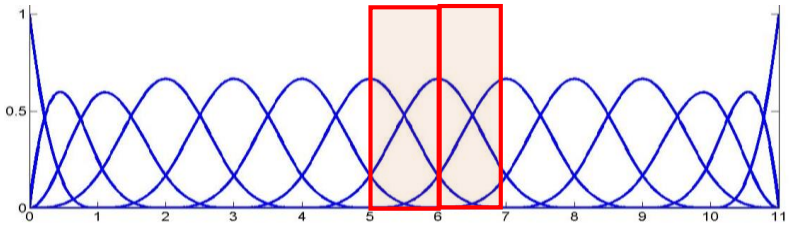
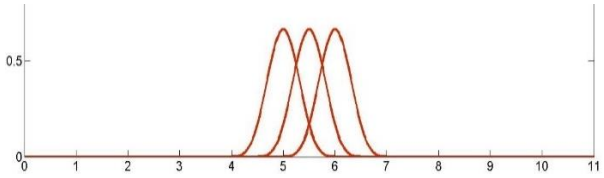
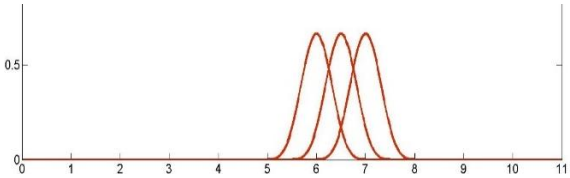


Figure 7.24.
Coarse basis and to be refined knot spans (shaded areas)
. (Iakovos Antonios and Karras Dimitrios , 2015)

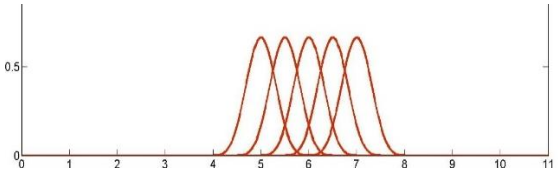
Two consecutive knot spans, [5, 6] and [6, 7], were chosen to be refined. For each of the refined knot spans, a set of three contracted support B-SPLines is introduced. As it can be seen in **Fig. 7.25** from the two sets introduced, one B-SPLine is common and should be taken into account just once as shown in **Fig. 7.25.c** .



(a) Children set for the first knot span



(b) Children set for the second knot span



(c) Union of the two previous children sets

Figure 7.25.
Corresponding contracted children are depicted of the basis depicted in the previous figure.
(Iakovos Antonios and Karras Dimitrios , 2015)

By overlaying the union of the contracted children, in the coarse basis, it is possible to find which coarse functions are being affected by the introduction of the finer set. Fine B-Splines have to be subtracted from the coarse ones when they share the same support as shown in **Fig. 7.26**.

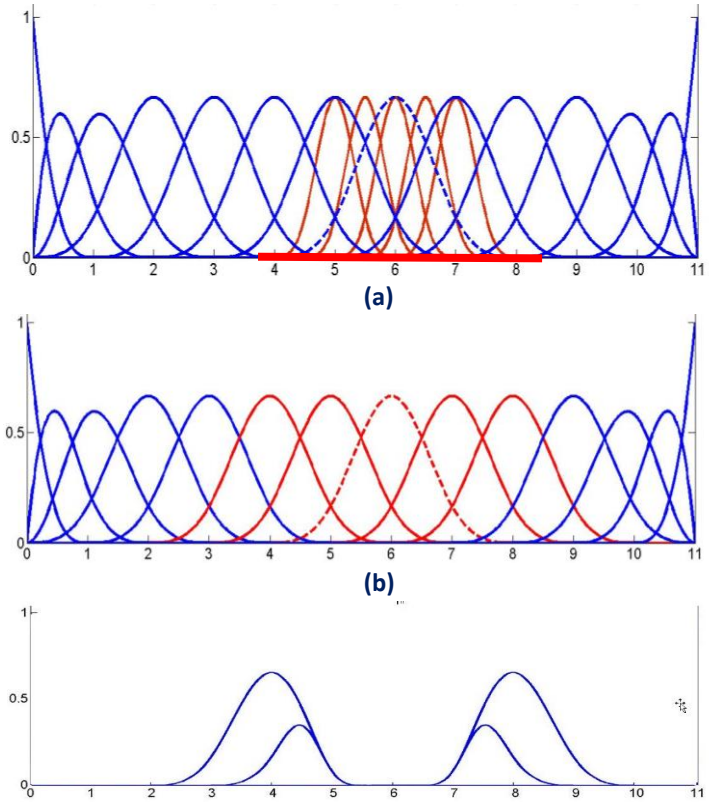


Figure 7.26.
(a) Overlaying of finer functions of level l-1 is being made.**(b)** Affected coarse B-Splines are being identified, as they interact with finer ones. **(c)**The result of subtraction of the finer B-Splines from the affected coarse ones
(Iakovos Antonios and Karras Dimitrios , 2015)

Next, we chose to refine knot span (5.5, 6) of level 2. Once again, we must introduce a set of finer B-Splines of contracted support in comparison to those of level 2. Thus, we introduce three new B-Splines that correspond to the depicted with gray knot span to be refined.

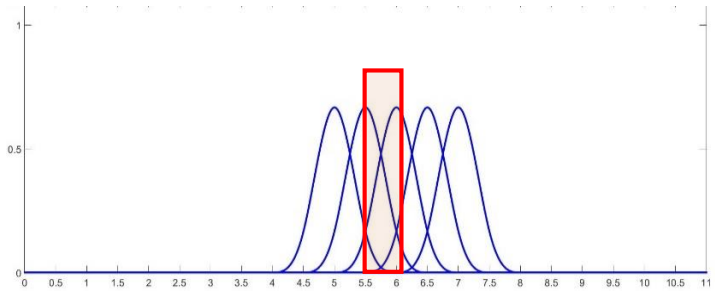


Figure 7.27.
Second level element to be refined
(Iakovos Antonios and Karras Dimitrios , 2015)

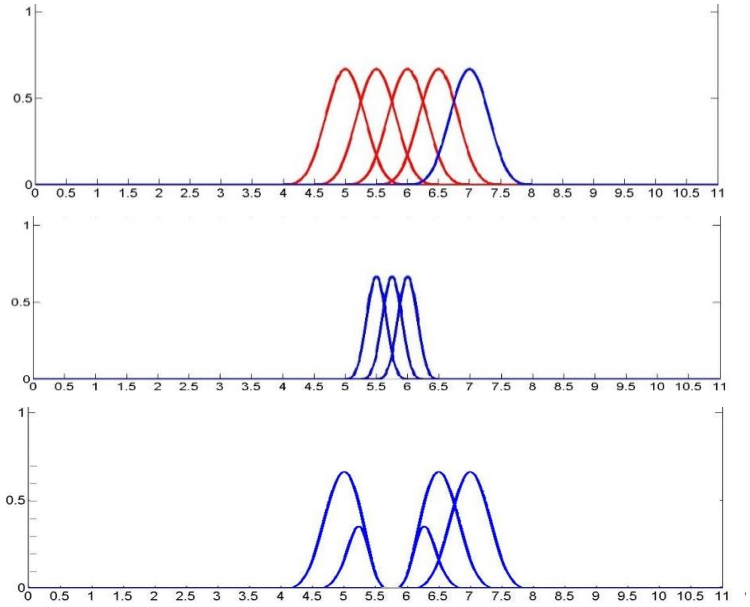


Figure7.28.
Identifying and subtraction procedure, between the second and third levels of the hierarchy.
(Iakovos Antonios and Karras Dimitrios , 2015)

Thus the final hierarchical basis for the refinement we applied will be as follows.

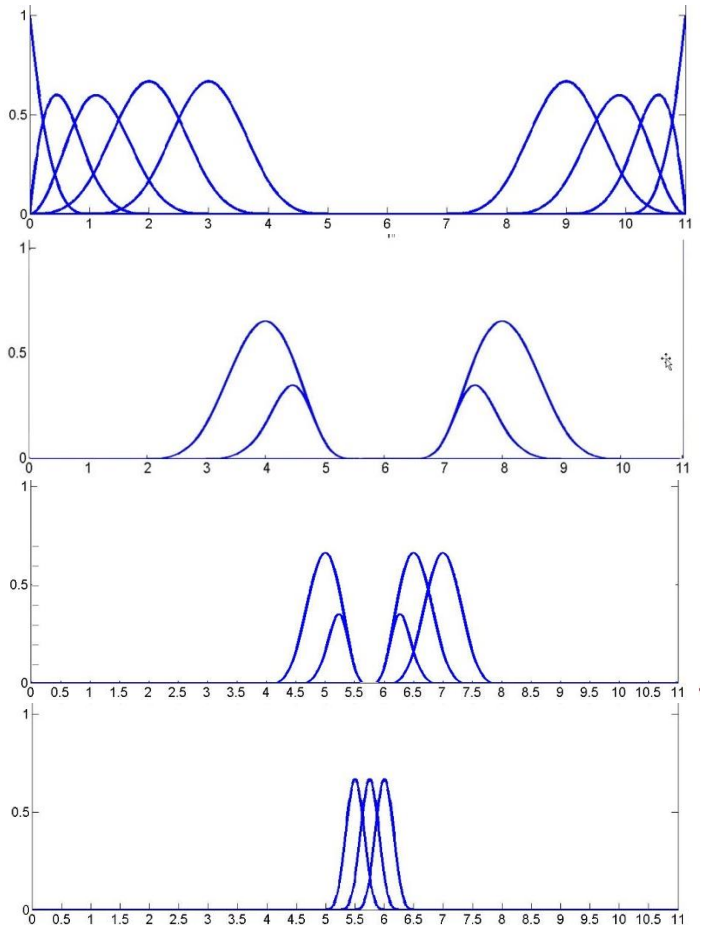


Figure 7.29.
Final hierarchical basis
(Iakovos Antonios and Karras Dimitrios , 2015)

7.4.3 Local Refinement in Multiple Dimensions

Assume there is a d-dimensional B-Spline parameter space (patch), which is generated by d univariate knot value vectors, one for each parametric direction by using the classical tensor product relation. An adaptive multi-level basis can be generated by successfully applying the 1D procedures analogously, since the tensor product structure allows for a decoupling of refinement operations in each parametric direction and subsequent d-dimensional assembly by multiplication. This is why it is very important to use local knot vectors for each multivariate B-Spline, as they will help the total implementation effort by producing easily the local knot vectors of the introduced children.

Now it is time to present the procedure for hierarchical local refinement in multiple dimension cases. The main idea remains the same, however adjustments are necessary in order to generalize accurately the 1D case.

- Initially, the adjusted nucleus operation referring to the refinement of one d-dimensional element of the mesh has to be introduced. Depending on the polynomial degree of each parametric direction (in this thesis it is considered the same for both axes), a suitable number of contracted B-Splines is chosen for each direction.

- The local increase of refinement follows the same concept as described previously, where the repetition of the nucleus operation in each overlay level ℓ and the repetition of hierarchical refinement with successively contracted B-Splines for the generation of the next overlay level $\ell+1$ leads to a multi-level B-Spline basis, which naturally accommodates adaptivity in all dimensions.

- Next step is the recovering linear independence. A linear combination of multivariate hierarchical B-Splines of the next level $\ell+1$ can represent B-Splines of the current level ℓ, in the sense of the multivariate two-scale subdivision relation. These linear dependencies need to be identified through the multi-level hierarchy and eliminated by a removal of the corresponding course level B-Splines. In analogy with the 1D case, this can be achieved by determining in each parametric direction, if the required p+2 contracted B-Splines of level $\ell+1$ exist in the hierarchical structure.

- Dirichlet boundary conditions can be incorporated weakly by variational methods, or strongly by a least squares fit of boundary basis functions. Homogeneous boundary conditions can be imposed strongly by removing all basis functions with support at the Dirichlet boundary.

8. LOCAL SOLUTION METHOD

8.1 INTRODUCTION

All the methodologies presented so far, had a common feature despite the mechanism used each time. They all calculated the global stiffness matrix for the final hierarchical basis repeating the solution of the whole problem for each refinement iteration. However, in classical FEA that is not always efficient, as calculating from scratch all the relations that connect the degrees of freedom for large scale problems is cost intensive.

It is clear that refining state variables and fast solving the equations that govern the examined problem, are two key issues in isogeometric analysis and more specifically in adaptive hierarchical refinement.

In computational mechanics, the time for solving a physical problem is consumed mostly at:

- Discretizing the definition domain.
- Formulating the stiffness matrix and then reverse it.
- Solving the system of the algebraic equations, which represent the PDE'S of the problem

For complex and large scale problems that contain millions of degrees of freedom, the size of the stiffness matrix is enormous and thus the equations need millions of iterations to be solved. So, the solving process may turn out to be a time-consuming step. For this reason, a more flexible approach for the solution of hierarchical refinement in isogeometric analysis was needed. In this direction Huazhong University of Science and Technology and the research team of Wu, Huang, Liu, Zuo proposed a new technique that demands special attention. It is a promising method of solving and not assembling the stiffness matrix in order to reduce the needed CPU time. Inspired by the hierarchical structure of the subdivided B-Splines and the decoupled idea from classical FEA, their proposal tries to separately construct and solve the equations of the individual hierarchical levels, instead of solving the global stiffness matrix.

The core idea of the proposed method is to divide the global stiffness matrix into several blocks according to the hierarchical refinement levels and to enforce boundary conditions in each level independently. These boundary conditions can be derived from the solution of the coarser level, in which the next finer level lies nested. This can happen, by utilizing the

coefficient matrices in order to describe the relations between the basis functions of different levels of the hierarchy. Finally the solution of the overall global problem is composed of the solutions for the individual levels.

8.2 COARSE, REFINED & BOUNDARY DOMAINS

In order to proceed to the local solution methodology Wu et al have discretized the parametric domain for one level of refinement in two smaller domains. These domains are two entities with different properties. Each of the subdomains consists of even smaller domains. More specifically, the whole parametric domain (patch) consists of the **coarse domain** and the **refined domain**. As the names give up, the first domain refers to the elements that have maintained their original size and the refined domain refers to the elements that have been contracted. Of course, the union of the two domains composes the complete mesh of level ℓ.

Coarse domain consists of the union of all elements of the initial-coarse domain of level ℓ, subtracting the union of all refined elements of the next level ℓ+1, which is the refined subdomain. The remaining lines/ areas/ volumes are the coarse domain. In general, coarse domain does not always refer to the coarsest domain of the mesh, but it refers to the coarser element set between two levels of refinement and analogously this applies also for the refined domain. Thus, by recursively applying hierarchical refinement we can conclude that, we will always get a set of one coarse and one refined domain for two consecutive levels ℓ and ℓ+1 of the hierarchy.

So, formally the coarse domain would be:

$$D_c^{\ell} = \bigcup_{i \in A_c^{\ell}} \mathrm{suppN}_i^{\ell} - \left\{ \bigcup_{j \in A_c^{\ell}} \mathrm{suppN}_j^{\ell+1} \cap \bigcup_{i \in A_c^{\ell}} \mathrm{suppN}_i^{\ell} \right\} \tag{8.1}$$

Additionally, refined domain can be further analyzed in two even smaller subdomains. For the previous discretization, the criterion was the distribution of the different size elements. Here, the discretization has as a criterion the intersections of the finer level ℓ+1 B-Splines with the coarser ones, as well as the union of their supports. More specifically, the refined domain is consisted of:

The **coupled domain**, which is defined as the support intersection of the basis functions that remain after removing the redundant ones between the two consecutive levels of the hierarchy. Coupled domain can be expressed as:

$$D_{\cap}^{\ell} = \bigcup_{i \in A_C^{\ell}} \text{suppN}_i^{\ell} \cap \bigcup_{j \in A_C^{\ell+1}} \text{suppN}_j^{\ell+1} \tag{8.2}$$

The **interior domain**, which is covered only by the supports of the refined basis functions of level $\ell+1$, but not by those of the previous level ℓ. So, the interior domain can be derived from:

$$D_{\text{int}}^{\ell} = \bigcup_{i \in A_C^{\ell+1}} \text{supp}N_i^{\ell+1} - D_{\cap}^{\ell} \tag{8.3}$$

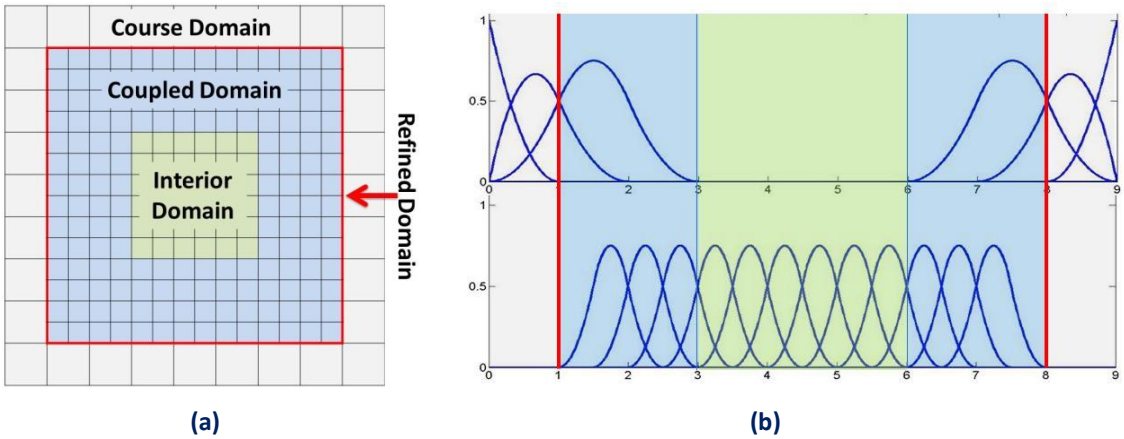


Figure 8.1.
Coarse, Coupled and Interior domains for **(a)** 1D and **(b)** 2D case respectively.
(Iakovos Antonios 2015)

In this sense, the traditional hierarchical refinement method is called global solution method, as the main goal is formulating and solving the stiffness matrix. The proposed method is respectively called local solution method.

The last but not least domain entity is the **boundary domain**. Actually, this domain is also defined by the intersection relations between the coarse and fine B-Splines of the mesh. Additionally, boundary domain is nested inside the coarse domain and in this sense it is a subdomain of the coarse domain. More specifically, boundary domain is defined by the HB-Splines of the hierarchical basis, which overlap over two levels. The support of them, which does not share support with the functions of the finer level, is the boundary domain.

Formally, boundary domain can be derived as:

$$D_B^{\ell} = \bigcup_{i \in A_C^{\ell}: \text{suppN}_i^{\ell} \in D_{\cap}^{\ell} \text{ and } \text{suppN}_i^{\ell} \in D_C^{\ell}} \text{suppN}_i^{\ell} - D_{\cap}^{\ell} \tag{8.4}$$

where the functions used are the active B-Splines of level ℓ have non-zero values over the coupled and the coarse domain simultaneously. Another expression for the boundary domain will be later discussed, as additional knowledge is needed in order to comprehend it in depth.

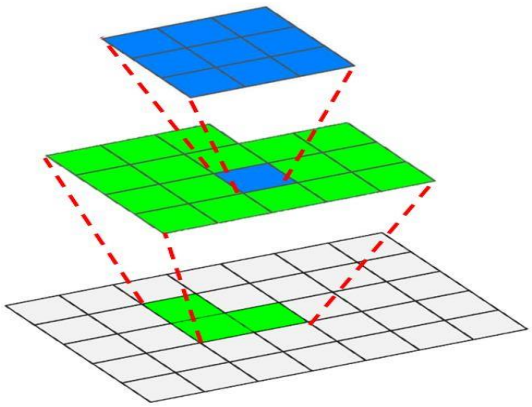


Figure 8.2
Hierarchy of the B-Splines in index space for a two level refinement.
(Karaiskos Giorgos , 2015)

8.3 Decoupling the Stiffness Matrix

In this section, the formulation, re-order and decoupling of the global stiffness matrix will be presented. More specifically, the local solution method will be explained step by step, in order to accelerate the solution of the algebraic system of equations.

Generally, the linear algebraic equations generated in IGA for a linear PDE with Galerkin method have the following classic form, no matter whether a uniform or hierarchical refinement is applied:

$$\{R\}_{1xN} = \left[K_{global}\right]_{NxN} \cdot \{u\}_{1xN} \tag{8.5}$$

where $[K_{global}]$ is the global sparse stiffness matrix and {R} is the vector related to boundary conditions and external loads.

In traditional hierarchical refinement, due to the nesting relations between the domains at different levels (8.9) can be decomposed into the block form for the case of one refinement level:

$$\begin{bmatrix} \left[K_{11}^1\right] & \left[K_{12}^1\right] & [0] \\ \left[K_{21}^1\right] & \left[K_{22}^1\right] & \left[K_{23}^1\right] \\ [0] & \left[K_{32}^1\right] & \left[K_{33}^1\right] \end{bmatrix} \bullet \begin{bmatrix} \left\{u_1^1\right\} \\ \left\{u_1^2\right\} \\ \left\{u_2^2\right\} \end{bmatrix} = \begin{bmatrix} \left\{R_1^1\right\} \\ \left\{R_1^2\right\} \\ \left\{R_2^2\right\} \end{bmatrix} \tag{8.6}$$

In (8.6):

- $\{u_1^1\}$ are the unknown displacements for the coarse domain D_C^1
- $\{u_1^2\}$ are the variables corresponding to coupled $D_\cap^1$ and
- $\{u_2^2\}$ the variables corresponding to interior D_{int}^1 domain.

Remember that the union of $D_\cap^1$ and D_{int}^1 domains composes the refined domain.

Moreover, the external load vectors R_1^1, R_1^2 and R_2^2 at the right hand side of (8.10) correspond to the aforementioned domains respectively.

Stiffness matrix for one refinement step in the coarse level 1 can be written as follows:

$$\begin{bmatrix} [K_{11}^1] & [K_{12}^1] \\ [K_{21}^1] & [K_{22}^1] \end{bmatrix} \bullet \begin{bmatrix} \{u_1^1\} \\ \{u_2^1\} \end{bmatrix} = \begin{bmatrix} \{R_1^1\} \\ \{R_2^1\} \end{bmatrix} \tag{8.7}$$

- $[K_{11}^1]$ refers to the coarse domain degrees of freedom
- $[K_{22}^2]$ refers to the refined domain degrees of freedom.
- $[K_{12}^1]$ and $[K_{21}^1]$ refer to the coupled domain degrees of freedom.

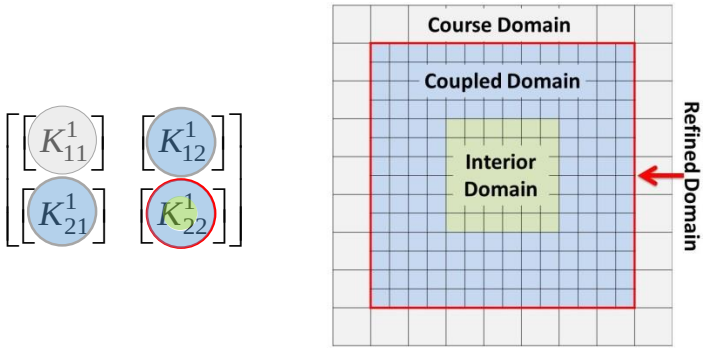


Figure 8.3.
The sub-matrices of the global stiffness matrix for one step of refinement are connected using the corresponding colors to the areas of degrees of freedom they represent.
(Iakovos Antonios 2015)

By using equation (8.7), we obtain initially results for the coarse level. In order to proceed further, the following assumption is necessary:

- If the numerical result errors are within the limits set beforehand**,** the corresponding sub-domain does not need to be recomputed.

If the resulting errors are exceeding the set boundary value, a global refinement may be needed firstly. This means that our original mesh is too coarse to implement the local solution method. However, in these cases, a parametric investigation may be needed in order to evaluate if it is worth to apply local refinement instead of a full tensor product h-refinement. This is of course a subject for further investigation.

So, after the refinement step, by ignoring the coarse domain results $\left\{u_1^1\right\}$, the fundamental equation (8.6) can be represented as:

$$\begin{bmatrix} [0] & [0] & [0] \\ \left[K_{21}^1\right] & \left[K_{22}^2\right] & \left[K_{23}^2\right] \\ [0] & \left[K_{32}^2\right] & \left[K_{33}^2\right] \end{bmatrix} \bullet \begin{bmatrix} \left\{u_1^1\right\} \\ \left\{u_1^2\right\} \\ \left\{u_2^2\right\} \end{bmatrix} = \begin{bmatrix} \left\{R_1^1\right\} \\ \left\{R_1^2\right\} \\ \left\{R_2^2\right\} \end{bmatrix} \Rightarrow \tag{8.8}$$

where the variables of the first row which correspond to the coarse domain are ignored. The equation (8.8) can be further reordered in order to achieve decoupling by writing the equation for the second row such as:

$$\begin{aligned} &\left[K_{21}^1\right]\cdot\left\{u_1^1\right\}+\left[K_{22}^2\right]\cdot\left\{u_1^2\right\}+\left[K_{23}^2\right]\cdot\left\{u_2^2\right\}=\left\{R_1^2\right\}\Rightarrow \\ &\left[K_{22}^2\right]\cdot\left\{u_1^2\right\}+\left[K_{23}^2\right]\cdot\left\{u_2^2\right\}=\left\{R_1^2\right\}-\left[K_{21}^1\right]\cdot\left\{u_1^1\right\} \end{aligned} \tag{8.9}$$

thus, by replacing (8.9) in (8.8) the decoupled matrix is derived as:

$$\begin{bmatrix} [0] & [0] & [0] \\ [0] & \left[K_{22}^2\right] & \left[K_{23}^2\right] \\ [0] & \left[K_{32}^2\right] & \left[K_{33}^2\right] \end{bmatrix} \bullet \begin{bmatrix} \left\{u_1^1\right\} \\ \left\{u_1^2\right\} \\ \left\{u_2^2\right\} \end{bmatrix} = \begin{bmatrix} \{0\} \\ \left\{R_1^2\right\}-\left[K_{21}^1\right]\cdot\left\{u_1^1\right\} \\ \left\{R_2^2\right\} \end{bmatrix} \tag{8.10}$$

where the blue matrices refer to the refined domain, and the red ones to the ignored coarse ones.

Now, by expanding the matrices of (8.7) in a 2x2 system form

$$\begin{aligned} &\left[K_{11}^1\right]\cdot\left\{u_1^1\right\}+\left[K_{12}^1\right]\cdot\left\{u_2^1\right\}=\left\{R_1^1\right\} \quad \text{(A)} \\ &\left[K_{21}^1\right]\cdot\left\{u_1^1\right\}+\left[K_{22}^1\right]\cdot\left\{u_2^1\right\}=\left\{R_2^1\right\} \quad \text{(B)} \end{aligned} \tag{8.11}$$

By solving (A) for u_2^1:

$$\begin{aligned}&\left[K_{11}^1\right]\cdot\left\{u_1^1\right\}-\left\{R_1^1\right\}=-\left[K_{12}^1\right]\cdot\left\{u_2^1\right\}\Rightarrow\\&-\left[K_{12}^1\right]^{-1}\left(\left[K_{11}^1\right]\cdot\left\{u_1^1\right\}-\left\{R_1^1\right\}\right)=\left\{u_2^1\right\}\end{aligned}\tag{8.12}$$

and then by substituting (8.12) in (B):

$$\left[K_{21}^1\right]\cdot\left\{u_1^1\right\}+\left[K_{22}^1\right]\cdot\left[-\left[K_{12}^1\right]^{-1}\left(\left[K_{11}^1\right]\cdot\left\{u_1^1\right\}-\left\{R_1^1\right\}\right)\right]=\left\{R_2^1\right\}\Rightarrow$$

$$\left[K_{21}^1\right]\cdot\left\{u_1^1\right\}-\left[K_{22}^1\right]\left[K_{12}^1\right]^{-1}\left[K_{11}^1\right]\cdot\left\{u_1^1\right\}+\left[K_{22}^1\right]\left[K_{12}^1\right]^{-1}\left\{R_1^1\right\}=\left\{R_2^1\right\}\Rightarrow$$

$$\left(\left[K_{21}^1\right]-\left[K_{22}^1\right]\left[K_{12}^1\right]^{-1}\left[K_{11}^1\right]\right)\cdot\left\{u_1^1\right\}=\left\{R_2^1\right\}-\left[K_{22}^1\right]\left[K_{12}^1\right]^{-1}\left\{R_1^1\right\}\Rightarrow$$

$$\left(\left[K_{22}^1\right]^{-1}\left[K_{21}^1\right]-\left[K_{12}^1\right]^{-1}\left[K_{11}^1\right]\right)\cdot\left\{u_1^1\right\}=\left[K_{22}^1\right]^{-1}\left\{R_2^1\right\}-\left[K_{22}^1\right]^{-1}\left[K_{22}^1\right]\left[K_{12}^1\right]^{-1}\left\{R_1^1\right\}\Rightarrow$$

$$\left(\left[K_{12}^1\right]\left[K_{22}^1\right]^{-1}\left[K_{21}^1\right]-\left[K_{11}^1\right]\right)\cdot\left\{u_1^1\right\}=\left[K_{12}^1\right]\left[K_{22}^1\right]^{-1}\left\{R_2^1\right\}-\left[K_{12}^1\right]\left[K_{12}^1\right]^{-1}\left\{R_1^1\right\}\Rightarrow$$

$$\left(\left[K_{11}^1\right]-\left[K_{12}^1\right]\left[K_{22}^1\right]^{-1}\left[K_{21}^1\right]\right)\left\{u_1^1\right\}=\left\{R_1^1\right\}-\left[K_{12}^1\right]\left[K_{22}^1\right]^{-1}\left\{R_2^1\right\}$$

(8.13)

So, finally by substituting (8.13) in the first row of (8.10), the global stiffness matrix can be expressed in its decoupled form as follows:

$$\begin{bmatrix}\left[K_{11}^1\right]-\left[K_{12}^1\right]\left[K_{22}^1\right]^{-1}\left[K_{21}^1\right] & [0] & [0]\\ [0] & \left[K_{22}^2\right] & \left[K_{23}^2\right]\\ [0] & \left[K_{32}^2\right] & \left[K_{33}^2\right]\end{bmatrix}\bullet\begin{bmatrix}\left\{u_1^1\right\}\\ \left\{u_1^2\right\}\\ \left\{u_2^1\right\}\end{bmatrix}=\begin{bmatrix}\left\{R_1^1\right\}-\left[K_{12}^1\right]\left[K_{22}^1\right]^{-1}\left\{R_2^1\right\}\\ \left\{R_1^2\right\}-\left[K_{21}^1\right]\cdot\left\{u_1^1\right\}\\ \left\{R_2^2\right\}\end{bmatrix}$$

(8.14)

It is obvious that each level can be now solved independently. The procedure explained is called by Wu et al local solution method.

Equation (8.14) can be represented in a more simple form:

$$\begin{bmatrix} [K^1] & \\ & [K^2] \end{bmatrix} \cdot \begin{bmatrix} \{u^1\} \\ \{u^2\} \end{bmatrix} = \begin{bmatrix} \{R^1\} \\ \{R^2\} \end{bmatrix} \tag{8.15}$$

where:

- $[K^1] = [K^1_{11}] - [K^1_{12}][K^1_{22}]^{-1}[K^1_{21}]$
- $\{u^1\} = \{u^1_1\}$
- $\{R^1\} = \{R^1_1\} - [K^1_{12}][K^1_{22}]^{-1}\{R^1_2\}$
- $[K^2] = \begin{bmatrix} [K^1_{22}] & [K^1_{23}] \\ [K^1_{32}] & [K^1_{33}] \end{bmatrix}$
- $\{u^2\} = \begin{bmatrix} \{u^2_1\} \\ \{u^2_2\} \end{bmatrix}$
- $\{R^2\} = \begin{bmatrix} \{R^2_1\} - [K^1_{21}] \cdot \{u^1_1\} \\ \{R^2_2\} \end{bmatrix}$

The (8.15) expression is valid for any two consecutive levels of refinement. For each level, the coarse and refined domains are changing constantly, as they represent entities of different levels in each iteration step.

Some very important features are being introduced by using the local solution method. This method does not provide an actual global stiffness matrix, but a number of smaller submatrices. The number of these submatrices, equals the number of hierarchical levels. The relations between stiffness matrices of consecutive levels need to be effectively handled. More specifically, index sorting of the basis functions from different levels and the generation of their coupling elements are necessary for the assembly of stiffness matrix.

8.4 ADJUSTING THE BOUNDARY CONDITIONS

The decoupling of equations for two consecutive levels of refinement has been theoretically approached. However, in order to achieve a practical implementation of the theory, it is necessary to introduce a procedure of properly imposing the boundary conditions to the nested spaces, with respect to the previous coarse domain. The subdivision property relation seems to be ideal for this purpose. The proposal of Bornemann et al in evaluating the two-scale coefficients seems to be legit for implementation.

In the local solution method, an intermediate Dirichlet boundary condition is introduced on the side of the refined domain to decouple the two hierarchical level equations. According to (8.14) and (8.15), the equation:

$$\{R\}^{\ell}_{1xN} = [K_{level}]^{\ell}_{NxN} \cdot \{u\}^{\ell}_{1xN}$$

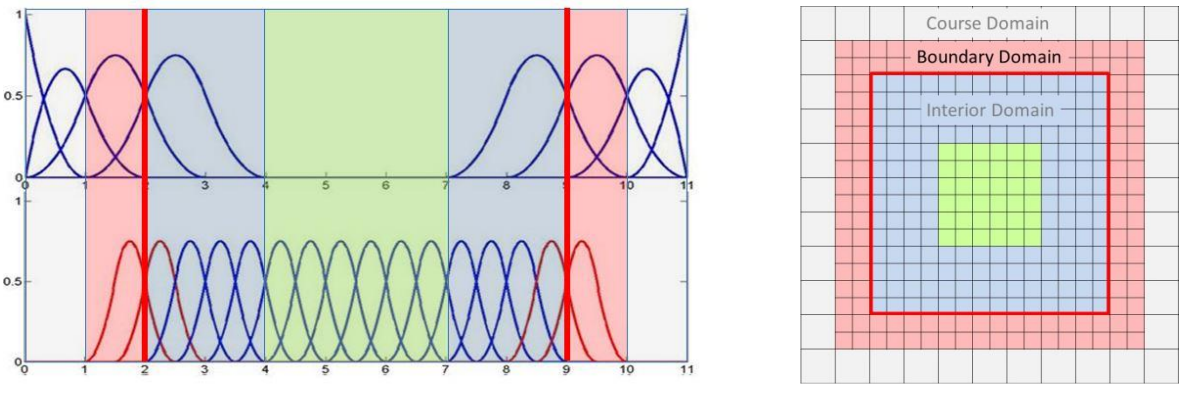
of level ℓ can be derived directly by using $\left\{u_1^{\ell-1}\right\}$, $\left\{R_1^{\ell}\right\}$ and $\left\{R_2^{\ell}\right\}$.

$\left\{u_1^{\ell-1}\right\}$ is already known from the solution of the previous level described in the following relation:

$$\{R\}^{\ell-1}_{1xN} = [K_{level}]^{\ell-1}_{NxN} \cdot \{u\}^{\ell-1}_{1xN} \tag{8.16}$$

$\left\{R_1^{\ell}\right\}$ and $\left\{R_2^{\ell}\right\}$, which are the external loads and boundary conditions from equation (8.6) for the coupled and interior domains are still unknown and have somehow to be found

It is chosen to obtain $\left\{R^{\ell}\right\}$ by imposing Dirichlet boundary conditions on the outer side of the coupled domain. This way leads boundary state variables of current level and results of the previous level to have the exact same values.



(a) (b)

Figure 8.4.
Boundary domain for **(a)** 1D and **(b)** 2D case respectively.
(Iakovos Antonios 2015)

For convenience of computation, just as in Bornemann et al algorithm, the boundary condition values are expressed with the bases at the lower level, because they originally have a coarser basis expression and the domain covered by these lower bases outside the refined domain is called boundary domain and is presented in **Fig. 8.4**.

Supposing a Spline C(ξ) in the boundary domain, it can be written as the combination of coarse and fine basis functions, such as:

$$C(\xi)=\left(u_1^{\ell-1}\right)^T N^{\ell-1}(\xi)+\left(u_1^{\ell}\right)^T N^{\ell}(\xi) \tag{8.17}$$

where $N^{\ell-1}$ and N^{ℓ} are the basis functions, which correspond to the coarse and the refined of level ℓ-1 and ℓ respectively.

From the definition of the boundary domain, we know that:

$$N^{\ell}(\xi)=0, \text{ on the outer side of } D_B$$

and thus, Dirichlet boundary conditions have the following values by using the subdivision matrix and the values that have been derived from the evaluation process of the hierarchy:

$$C(\xi)=\left(u_1^{\ell-1}\right)^T\cdot N^{\ell-1}(\xi)=\left(u_1^{\ell-1}\right)^T\cdot T^{\ell}\cdot N^{\ell}(\xi)=\left(u_b^{\ell}\right)^T\cdot N^{\ell}(\xi) \tag{8.18}$$

u_b^{ℓ} are the control points obtained by transforming $u_1^{\ell-1}$ onto level ℓ. Generating the stiffness equation (8.6) for the refined domain exclusively, using the basis functions at level ℓ, leads to the following form:

$$\begin{bmatrix}[I] & [\emptyset] & [\emptyset]\\ [K_{21}^{\ell}] & [K_{22}^{\ell}] & [K_{23}^{\ell}]\\ [\emptyset] & [K_{32}^{\ell}] & [K_{33}^{\ell}]\end{bmatrix}\begin{bmatrix}\{u_b^{\ell}\}\\ \{u_1^{\ell}\}\\ \{u_2^{\ell}\}\end{bmatrix}=\begin{bmatrix}\{u_b^{\ell}\}\\ \{R_1^{\ell}\}\\ \{R_2^{\ell}\}\end{bmatrix}\Rightarrow$$

$$\begin{bmatrix}[I] & [0] & [0]\\ [K_{21}^{\ell}] & [K_{22}^{\ell}] & [K_{23}^{\ell}]\\ [0] & [K_{32}^{\ell}] & [K_{33}^{\ell}]\end{bmatrix}\begin{bmatrix}\{u_b^{\ell}\}\\ \{u_1^{\ell}\}\\ \{u_2^{\ell}\}\end{bmatrix}=\begin{bmatrix}\{u_b^{\ell}\}\\ \{R_1^{\ell}\}\\ \{R_2^{\ell}\}\end{bmatrix} \tag{8.19}$$

$$\begin{bmatrix} \left[K_{22}^{\ell}\right] & \left[K_{23}^{\ell}\right] \\ \left[K_{32}^{\ell}\right] & \left[K_{33}^{\ell}\right] \end{bmatrix} \begin{bmatrix} \left\{u_1^{\ell}\right\} \\ \left\{u_2^{\ell}\right\} \end{bmatrix} = \begin{bmatrix} \left\{R_1^{\ell}\right\} - \left[K_{21}^{\ell}\right]\left\{u_b^{\ell}\right\} \\ \left\{R_2^{\ell}\right\} \end{bmatrix} \tag{8.20}$$

Actually, vector $\left\{u_b^{\ell}\right\}$ represents the boundary value on the inner side of the boundary domain with a set of control points:

$$\left\{u_b^{\ell}\right\} = \left[u_{b,1}^{\ell}, u_{b,2}^{\ell}, \ldots, u_{b,\mathrm{m}}^{\ell}\right]^T \tag{8.21}$$

which do not belong to the control points $\left\{u_1^{\ell}\right\}$ of the coupled domain, but can be derived from $\left\{u_1^{\ell-1}\right\}$ (control points of the coarse domain) as:

$$\left\{u_{b,i}^{\ell}\right\} = a_i \cdot u_{b,i}^{\ell-1} + \left(1 - a_i\right) \cdot u_{b,i}^{\ell-1} \tag{8.22}$$

where:

$$\left\{u_1^{\ell-1}\right\} = \left[u_{b,1}^{\ell-1}, u_{b,2}^{\ell-1}, \ldots, u_{b,\mathrm{m}}^{\ell-1}\right]^T \tag{8.23}$$

are the control points of the upper (coarser) level that express the solution values on the boundary. Moreover a_i are the coefficients from the classical knot insertion in NURBS, which give the Cartesian coordinates for the newly introduced finer functions. This way we can find the finer control point coordinates, in order to use them to decouple the equations.

Since $\left\{u_b^{\ell}\right\}$ are used to define the intermediate Dirichlet boundary condition, the boundary domain can be written as:

$$D_B = \bigcup_i \mathrm{supp}\left(N_{b,i}^{\ell}\right) - D_{\cap} \tag{8.24}$$

where $N_{b,i}^{\ell}$ are the basis functions associated to the set of control points $\left\{u_b^{\ell}\right\}$.

Due to the characteristic of non-interpolation basis functions $N_{b,i}^{\ell}$ are not arranged in a sequence like the way in the traditional FEA implementations. The width W_B of the boundary domain is equal to p. Thus:

$$Width\left(B_D\right) = W_B = p \tag{8.25}$$

Practically this means that boundary domain will always cover p elements around the perimeter of the refined domain. In other words, W is the number of the control points along the normal direction of the boundary.

When two or more discrete areas are being refined, it is obvious that the same number of boundary domains is created. However, the boundary domains may be related in three ways: disjoined, adjacent and overlapped as shown in **Fig. 8.5**.

- For the disjoined domains, each refined domain has its own independent boundary domain and thus has a separate equation that can be created and solved independently.
- For the adjacent and overlapped boundary domains with their width W not greater than p, the refined domains have a part of common boundary and thus only one equation.

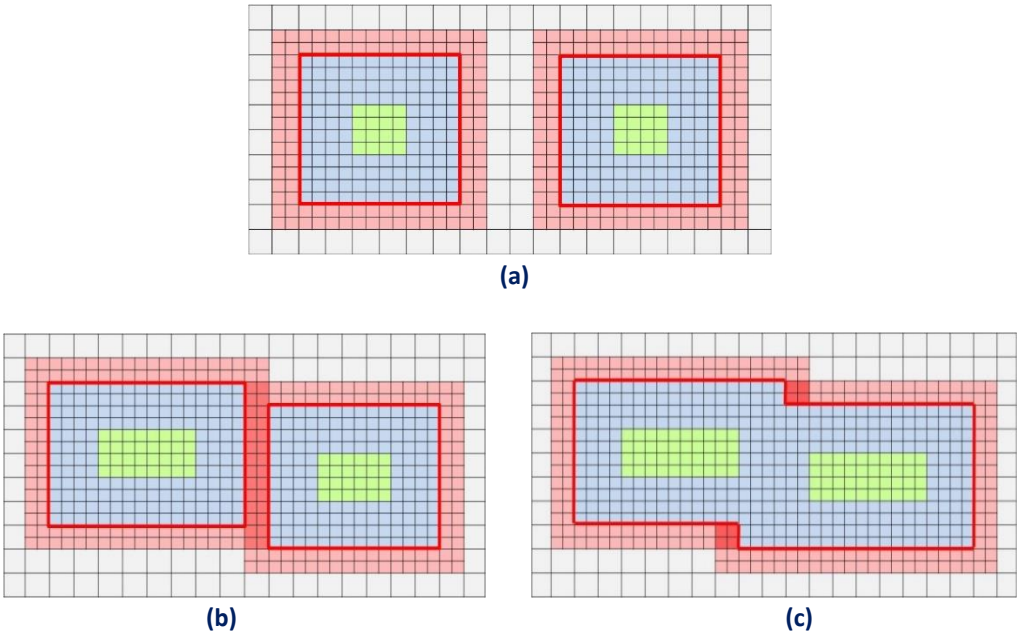


Figure 8.5.
(a) Disjointed boundary domains, **(b)** Adjacent boundary domains, **(c)** Overlapped boundary domains. (Iakovos Antonios 2015)

8.5 CONCLUSIONS

To summarize, the Wu et al proposal utilized the hierarchical refinement procedure based on the concept of B-Splines along with the subdivision property in order to provide a local solution nature for each level. In order to do that, they used the transformation matrix S between the basis functions of the refined and its previous level. Additionally, the local solution method was proposed by exploiting the characteristic of hierarchical refinement structure and applied an error indicator to achieve the adaptive nature, which seems to be crucial. The refinement for non-uniform representation turns out to be very flexible, because it is suitable for parameter domains refined with arbitrary node intervals and node multiplicity.

The local solution method appears to be applicable to hierarchical refinement in IGA. It has almost the same accuracy as the traditional global solution method, while it bears less computational cost for the same degrees of freedom

9. ADAPTIVE REFINEMENT ALGORITHM

9.1 INTRODUCTION

The adopted by this thesis theoretical and algorithmic approach for adaptive hierarchical refinement in isogeometric analysis will be presented in the present chapter. The theoretical aspects described in previous chapters were mainly focused on the 1D case. It was chosen to describe and explain different adaptive refinement methods in the frame of one-dimensional problems, as the generalization to multiple dimensions is quite straightforward. In this chapter elements of the aforementioned hierarchical refinement methods will be combined to introduce a simple approach for dealing with 2D problems.

It is worth to mention that all the proposed methods lead to hierarchical bases that share the same properties. Two of them are for example linear independence and partition of unity. Another important aspect is their form. All the produced bases can be divided in characteristic areas such as the ones defined in the local solution method. This means that they can be easily combined to solve the final problem. Optimizing the hierarchical basis creation and combining it with the local solution method could just be a possible combination. This combination could be examined for its overall efficiency in solving problems using the adaptive hierarchical refinement method.

But before proceeding to a combination, such as the one presented previously, it is of crucial importance to investigate the parametric comparison between classical hierarchical refinement and uniform refinement. In this sense some 2D examples will be analyzed with the two main refinement methods (adaptive hierarchical and uniform refinement) so as to compare their efficiency, opening the way for new research topics concerning the specific refinement approach.

9.2 THESIS' APPROACH TO HIERARCHICAL REFINEMENT

9.2.1 General procedure

As it is already said, an algorithm of adaptive hierarchical refinement will be presented in this thesis. This algorithm refers to solution of 2D problems. But it can be easily used for

one-dimensional and three-dimensional problems respectively. The basic procedure still remains the same.

The first step is to solve the problem using its initial parametrization. The initial solution is now used as the input to the iterative procedure of adaptive hierarchical refinement. Each iteration can be characterized by three main procedures.

The first one is the validation of the existing solution and the identification of areas in need of refinement. This procedure is in fact the most important of all, as it is always the key aspect in an adaptive algorithm. To validate solution's efficiency and to identify regions with higher error, an efficient and easy to handle error estimator is needed.

The second procedure can be named basis formulation. In this part the hierarchical basis is constructed. It will be explained thoroughly in the following units how this can be achieved. In few words in this step the approximation space is enriched in respect to certain properties (linear independence, partition of unity) with finer shape functions, in order to increase solution's precision. The result of the hierarchical basis construction is the formulation of the corresponding stiffness matrix.

The third important procedure of the iteration step is the solution of the enriched problem. The precision of the results is the one that defines if another iteration is needed in case the desired convergence is not achieved.

The general procedure is graphically presented in the following figure.

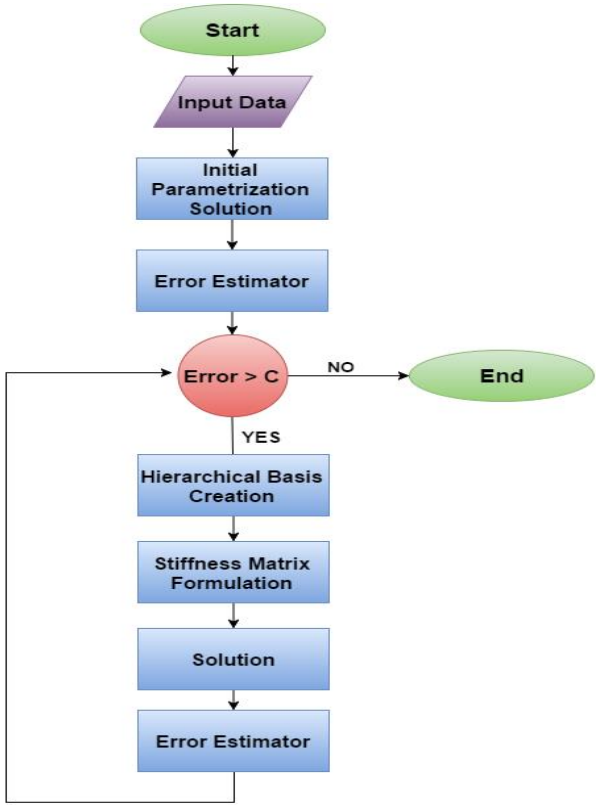


Figure 9.1.
Flow chart of adaptive hierarchical refinement algorithm.
(Iakovos Antonios 2015)

9.2.2 Selected entities for refinement

To begin with, it is well known that classical FEM adopts an element viewpoint for all its procedures and that refinement consists of the geometric division of elements. Things are not as straightforward as in FEM when it is about isogeometric analysis, as the definition of elements actually refers to knot spans. Every element of isogeometric analysis shares shape functions with its neighbor elements. So it is crucial to decide where refinement should be applied. Refinement could be applied directly to shape functions or indirectly to them by locating elements that need refinement. In the second case candidate shape functions for refinement are those that interact with the chosen elements. It is obvious that this way leads to loss of the desired local nature of hierarchical refinement.

In order to define the entities that need to be refined there has to be a way to evaluate their corresponding error. For this reason an efficient and easy to handle error estimator is necessary. Error estimators in isogeometric analysis are still a topic of research. Error can be evaluated on the viewpoint of elements or on the viewpoint of shape functions. The entity where the error is estimated is the one that determines the way refinement is applied. It is obvious that error estimator has a key role in adaptive refinement as it is an indicator for the iterative refinement procedure. In this thesis the kind of the used error estimator is the one that refers to element entities. The chosen error estimator will be thoroughly presented in one of the following units.

So the entities chosen for refinement are elements. It is now important to decide which shape functions to refine. For the algorithm used in this thesis if an element is marked for refinement all shape functions that interact with this element are refined. The explained shape function selection is presented in **Fig. 9.2**. The drawback of this selection is that if all interacting shape functions are chosen, the refinement area is far from being characterized local. Due to the overlapping of shape functions in isogeometric refinement is spread to elements that are not marked by the error indicator increasing the number of degrees of freedom. Fortunately there are some ways to receive results of the desired accuracy by introducing fewer degrees of freedom, as it will be shown clearly in the applications presented in the next pages.

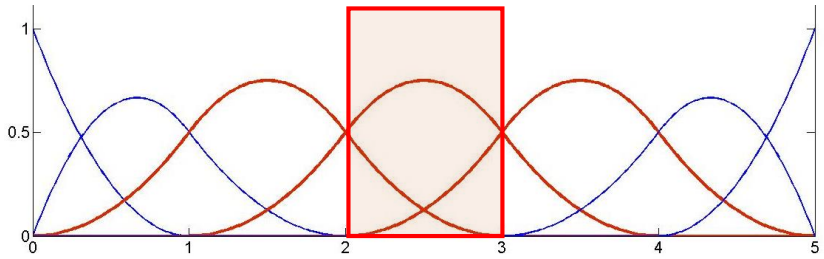


Figure 9.2.

Example of basis function selection for refinement. The marked knot span for refinement is [2,3). All the basis functions that intersect with this knot span are to be replaced by finer basis functions. It is obvious that these basis functions are very spread, not satisfying the desired local nature.
(Iakovos Antonios 2015)

9.2.3 Hierarchical level connection relations

Following the first step of refinement, which is the selection of the shape functions that have to be refined, it is time to explain how the basis is enriched with finer basis functions. For the formation of enriched hierarchical bases, subdivision property is used. Each marked shape function is replaced with its $(p+2)^d$ children shape functions that can represent it as linear combination. In order to satisfy this last condition which is an extension of partition of unity, each subsidiary shape function is accompanied by a scale factor derived from the subdivision rule for uniform basis functions and by the use of some other coefficients for non-uniform cases. The second type of coefficients was also presented in **Chapter 6**. To remind them there exist relations for polynomial degree p=2 and polynomial degree p=3 as presented below.

Coefficients for quadratic basis function defined by a span $\left[\xi_1,\xi_2,\xi_3,\xi_4\right]$:

$$\alpha_1=\frac{\xi_2-\xi_1}{2\left(\xi_3-\xi_1\right)} \qquad \alpha_2=1-\frac{\xi_3-\xi_2}{2\left(\xi_3-\xi_1\right)} \qquad \alpha_3=1-\frac{\xi_3-\xi_2}{2\left(\xi_4-\xi_2\right)} \qquad \alpha_4=1-\frac{\xi_4-\xi_3}{2\left(\xi_4-\xi_2\right)}$$

As for the cubic basis function with a local knot vector $\left[\xi_1,\xi_2,\xi_3,\xi_4,\xi_5\right]$ the coefficients can be identified in the similar way:

$$\alpha_1=\frac{\xi_2-\xi_1}{2\left(\xi_3-\xi_1\right)}\frac{\xi_3+\xi_2-2\xi_1}{2\left(\xi_4-\xi_1\right)} \qquad \alpha_2=\frac{\xi_3+\xi_2-2\xi_1}{2\left(\xi_4-\xi_1\right)}$$

$$\alpha_3=\frac{\xi_4-\xi_3}{2\left(\xi_4-\xi_2\right)}\frac{-\xi_1+\xi_2+\xi_3-\xi_4}{2\left(\xi_4-\xi_1\right)}+\frac{\xi_3-\xi_2}{2\left(\xi_4-\xi_2\right)}\frac{\xi_2-\xi_3-\xi_4+\xi_5}{2\left(\xi_5-\xi_2\right)}+\frac{3}{4}$$

$$\alpha_4=\frac{-\xi_4-\xi_3+2\xi_5}{2\left(\xi_5-\xi_2\right)} \qquad \alpha_5=\frac{+\xi_4+\xi_3-2\xi_5}{2\left(\xi_5-\xi_2\right)}\frac{\xi_5-\xi_4}{2\left(\xi_5-\xi_3\right)}$$

Note that for uniform local knot value vectors the above relations yield the same scale factors as the ones derived from the subdivision property. If for example the support of a quadratic basis function is [0, 1, 2, 3] the corresponding relations give α_1=0.25, α_2=0.75, α_3=0.75, α_4=0.25 which coincide with subdivision scale factors.

So the refinement of basis functions can be described by the following equation.

$$N_{i,p}^{k}(\xi)=[\alpha][N]=\sum_{j=0}^{p+1}a_j N_{m,p}^{k+1}(\xi) \tag{9.1}$$

The knot values used in this thesis are open knot value vectors. Thus the multiplicity at the two end knot points is p+1 and the related basis functions are discontinuous. These

boundary basis functions have only two children and their coefficients are 1 and 0.5 respectively whatever their degree is. Their transformational expression can be written as follows:

$$N_1(\xi) = [N_1^{k+1}(\xi) \quad N_2^{k+1}(\xi)] \cdot \varphi$$
$$N_n(\xi) = [N_m^{k+1}(\xi) \quad N_{m-1}^{k+1}(\xi)] \cdot \varphi \tag{9.2}$$

where $\phi=[1\ ,\ 0.5]^T$ and m is the index of the basis function at the right side.

Having already introduced the needed scale factors, it is very important to relate maternal and subsidiary shape function IDs. In this way subsidiaries of a consecutive finer level can be found. To simplify this connection it is vital to introduce relations for one-dimensional problems and then expand them to n-dimensional problems. It is known from the subdivision rule that every function can be described as linear combination of p+2 subsidiary shape functions. The purpose is to define the global ID of children shape functions when the global ID of the maternal shape function is known. Every shape function can be written as N_i^l where i is its global ID and *l* is its corresponding level. Every shape function of level *l* with ID i has p+2 children of level *l*+1 with IDs $i_s=[2i_m-(p+1),\ 2i-p,\ \ldots.,\ 2i]$, where i_s are the children's IDs and i_m is the ID of the maternal basis function.

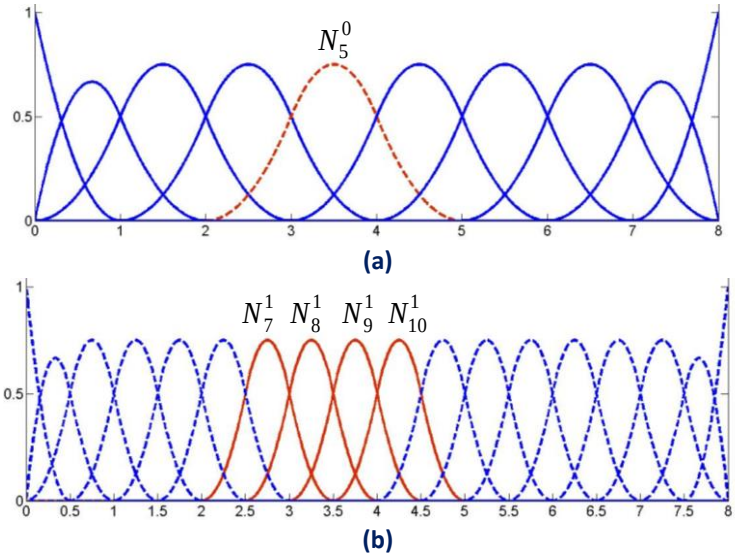


Figure 9.3.
Maternal basis function and its corresponding finer level children ID relations.
(Iakovos Antonios 2015)

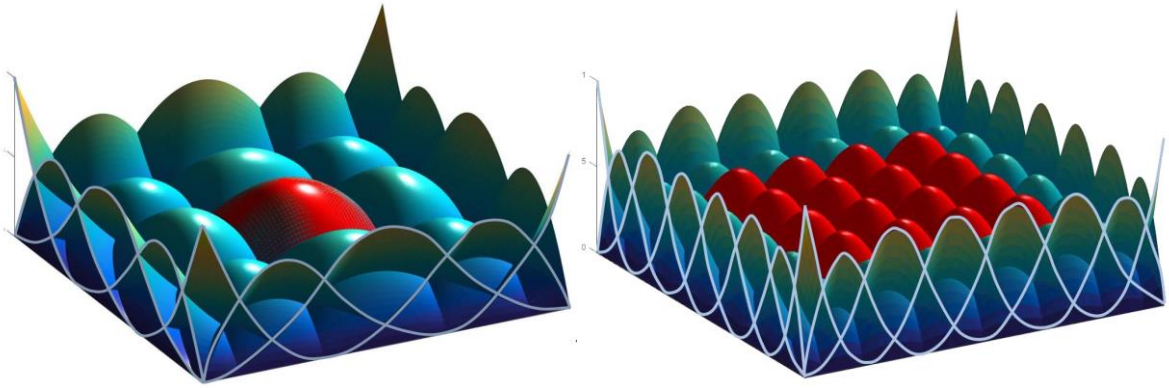
The proposed ID relation can be validated by applying it on the example presented in **Fig. 9.3**. The polynomial degree of the basis functions is p=2, thus the number of children corresponding to N_5^0 are p+2=4. Using the proposed relation children's IDs are

$$[2\cdot 5-(2+1),\quad 2\cdot 5-2,\quad 2\cdot 5-1,\quad 2\cdot 5] =$$
$$[7,\quad 8,\quad 9,\quad 10]$$

So the corresponding children to maternal basis function N_5^0 are $[N_7^1 \quad N_8^1 \quad N_9^1 \quad N_{10}^1]$ as shown in **Fig. 9.3.b**.

Using the above relations it is possible that some children ID's can be non-positive or greater than the total number of fine basis functions. This problem can be observed for the first and last p basis functions respectively. In this case the unacceptable values are discarded along with their corresponding scale factors that always are equal to zero according to their calculation relations.

This procedure can be extended in n-dimensional cases by defining the parametric IDs of shape functions. From the parametric IDs of maternal shape functions derive the parametric IDs of subsidiary shape functions and consequently their global ID



(a) (b)

Figure 9.4.
Maternal basis function and its corresponding finer level children ID relations.
(Iakovos Antonios 2015)

In **Fig. 9.4.a** shape function N_{13}^0 is chosen for refinement. Its parametric IDs are (3,3). Applying the relations for each parametric axis the parametric IDs of subsidiary basis functions can be derived. Their parametric IDs are [3, 4, 5, 6] for both ξ and η axis. Combining the responding parametric IDs the subsidiary local IDs can be derived.The subsidiary functions corresponding to N_{13}^0 are :

$[N_{19}^1 \quad N_{20}^1 \quad N_{21}^1 \quad N_{22}^1 \quad N_{27}^1 \quad N_{28}^1 \quad N_{29}^1 \quad N_{30}^1 \quad N_{35}^1 \quad N_{36}^1 \quad N_{37}^1 \quad N_{38}^1 \quad N_{43}^1 \quad N_{44}^1 \quad N_{45}^1 \quad N_{46}^1]$

and are depicted in **Fig. 9.4.b** with red color.

The relations between two consecutive levels are now well defined, so all the subsidiary shape functions can be easily identified. Maternal shape functions are then replaced by them. In this part of the procedure it is time to check if linear independence is satisfied. In case any shape function in level *l* can be fully described as linear combination of the introduced finer shape functions of level *l*+1, it has to be marked as redundant and totally removed from the hierarchical basis. This can be checked by comparing their element supports.

9.2.4 Evaluation of hierarchical control point coefficients

One key aspect in isogeometric analysis is that even when meshes are refined geometry always remains the same. Evaluation of control point coefficients is applied taking into account this restriction. The control point positions have to be updated and placed in such positions, so that their resulting geometry is identical to the initial one.

Geometry on level *l* is represented by the following equation

$$u^{l}(\xi,\eta,\zeta)=\left[N^{l}(\xi,\eta,\zeta)\right]^{T}\cdot\left\{u^{l}\right\} \tag{9.3}$$

Respectively in level *l*+1

$$u^{l+1}(\xi,\eta,\zeta)=\left[N^{l+1}(\xi,\eta,\zeta)\right]^{T}\cdot\left\{u^{l+1}\right\} \tag{9.4}$$

To satisfy the restriction of the same geometry before and after refinement

$$\begin{gathered}u^{l}(\xi,\eta,\zeta)=u^{l+1}(\xi,\eta,\zeta)\Leftrightarrow\\ \left[N^{l}(\xi,\eta,\zeta)\right]^{T}\cdot\left\{u^{l}\right\}=\left[N^{l+1}(\xi,\eta,\zeta)\right]^{T}\cdot\left\{u^{l+1}\right\}\end{gathered} \tag{9.5}$$

Matrix [S] is now introduced. S is a matrix with n_s rows and n_m columns, where n_s is the number of subsidiaries and n_m is the number of maternal shape functions. It has been proven that:

$$\begin{gathered}\left[N^{k}(\xi,\eta,\zeta)\right]=[S]^{T}\left[N^{k+1}(\xi,\eta,\zeta)\right]\Rightarrow\\ \left[N^{k}(\xi,\eta,\zeta)\right]^{T}=\left[N^{k+1}(\xi,\eta,\zeta)\right]^{T}[S]\end{gathered} \tag{9.6}$$

By substituting $[N^{k}]^{T}$ in the previous equation

$$\begin{gathered}\left[N^{l+1}(\xi,\eta,\zeta)\right]^{T}\cdot\left\{u^{l+1}\right\}=\left[N^{l+1}(\xi,\eta,\zeta)\right]^{T}[S]\cdot\left\{u^{l}\right\}\Rightarrow\\ \left\{u^{l+1}\right\}=[S]\cdot\left\{u^{l}\right\}\end{gathered} \tag{9.7}$$

In the simplest case that a one-dimensional maternal basis function is replaced by its p+2 children, transformation matrix [S] is actualy a vector with p+2 rows.

$$[S]=\{S_1\quad S_2\quad\cdots\quad S_{p+2}\}^{T} \tag{9.8}$$

In this case control point coefficients are actually calculated as the maternal control point coefficient multiplied by the corresponding scale factors that derived from the subdivision property.

Another possibility is that a subsidiary is child of more than one maternal basis functions. Then its control point coefficients can be calculated by summing up the weighted maternal control point coefficients, where subdivision scale factors are again used as corresponding weights. Such an example is depicted in the transformation matrix [S] of **Fig. 9.5.** N_5^2 for example is a common child of N_3^1 and N_4^1. Its coefficients are the sum of the N_3^1 corresponding coefficients multiplied with 0.75 and the N_4^1 coefficients multiplied with 0.25.

	N^1_1	N^1_2	N^1_3	N^1_4	N^1_5	N^1_6
N^1_1	1	0	0	0	0	0
N^1_2	0	1	0	0	0	0
N^2_3	0	0	0.25	0	0	0
N^2_4	0	0	0.75	0	0	0
N^2_5	0	0	0.75	0.25	0	0
N^2_6	0	0	0.25	0.75	0	0
N^2_7	0	0	0	0.75	0	0
N^2_8	0	0	0	0.25	0	0
N^1_5	0	0	0	0	1	0
N^1_6	0	0	0	0	0	1

Figure 9.5.
Typical transformation matrix [S] for quadratic one-dimensional basis functions.
(Iakovos Antonios 2015)

9.2.5 Stiffness matrix formulation

The formulation of stiffness matrix follows the procedure explained in **Chapter 3**

Just to remind 2D stiffness matrix is calculated by the following equation

$$\underset{(2N\times 2N)}{[K]} = \int_{\xi_0}^{\xi_{n+p+1}} \int_{\eta_0}^{\eta_{m+q+1}} \underset{(2N\times 3)}{\left[B(\xi,\eta)\right]^T} \cdot \underset{(3x3)}{[E]} \cdot \underset{(3\times 2N)}{\left[B(\xi,\eta)\right]} \cdot t \cdot \det[J]\, d\eta d\xi \tag{9.10}$$

Numerical integration procedures for ξ,η lead to integration of tensor product Gauss points can be applied as follows :.

$$\underset{(2N\times 2N)}{[K]} = \sum_{i=1}^{GP\xi} \sum_{j=1}^{GP\eta} \underset{(2N\times 3)}{\left[B(\xi_i,\eta_j)\right]^T} \cdot \underset{(3x3)}{[E]} \cdot \underset{(3\times 2N)}{\left[B(\xi_i,\eta_j)\right]} \cdot t \cdot \det[J] \cdot w_i^{GP\xi} \cdot w_j^{GP\eta} \tag{9.11}$$

The equations remain the same for hierarchical refinement. The only changed variable is the position of Gauss points. The positions of Gauss points per element are pretty much predefined. But it is necessary to consider that hierarchical meshes are a union of different level domains. In areas where only course shape functions exist, their values are evaluated at the corresponding course Gauss points. The same goes in areas where only fine level

shape functions exist. Their values are calculated at the fine Gauss points. The question that arises is what happens in areas where both course and fine shape functions exist. In this case it is necessary to consider that all the elements that form the support of fine level shape functions are subdivided. Gauss points are always defined per element. So, in these special areas where elements are subdivided, course and fine shape functions values are both calculated at the introduced fine Gauss points. In general it can be said that integration per element is always applied on Gauss points that correspond to the level of the finest shape functions that interact with each element.

It has to be mentioned that the number of total Gauss points is increased exponentially when hierarchical refinement is applied. This drawback could be overcome by using other collocation methods. Collocation methods are still a running research issue, very handful to be incorporated in adaptive algorithms in isogeometric analysis, so as to decrease the computational cost.

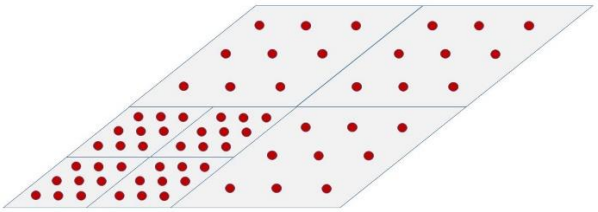


Figure 9.6.
Gauss points on hierarchical mesh. All values calculated on elements are defined in Gauss points that correspond to the level of the finer basis functions interacting with each element.
(Iakovos Antonios 2015)

9.2.6 Error estimator

It wouldn't be excessive to say that error estimator is the most crucial part of an adaptive algorithm. In such an algorithm adaptive refinement has to be automatic based on error control. There is need of an objective quantity used as criteria for determining whether a region should be further refined or not.

Firstly it has to be reminded what solution error is. In IGA, similar to FEA, the error *e* is defined as the difference of the exact solution $\tilde{u}$ and the approximate results u.

$$e = \tilde{u} - u$$

The problem is that the exact value $\tilde{u}$ is not known beforehand. So, an error indicator has to be introduced. A simple gradient based error indicator ε is used for this thesis, since its aim is to capture steep gradients in the solution. For each element the local error ε_e can be estimated as:

$$\varepsilon_e = \frac{1}{V_e}\left(\int_{\Omega_e} |\nabla u|^2 \, d\Omega_e\right)^{1/2} \tag{9.12}$$

where u is the solution field evaluated from the current mesh of hierarchical level *l*. The indicator is evaluated over the domain Ω_e of each element and subsequently normalized with respect to the corresponding element volume V_e.

The solution field is defined as follows:

$$u = \begin{Bmatrix} u(x,y) & v(x,y) \end{Bmatrix}^T \tag{9.13}$$

The gradient of the solution field is equal to

$$\nabla u = \begin{bmatrix} \frac{\partial}{\partial x}u(x,y) & \frac{\partial}{\partial y}u(x,y) \\ \frac{\partial}{\partial x}v(x,y) & \frac{\partial}{\partial x}v(x,y) \end{bmatrix} = \begin{bmatrix} u_{,x}(x,y) & u_{,y}(x,y) \\ v_{,x}(x,y) & v_{,y}(x,y) \end{bmatrix} \tag{9.14}$$

so

$$\left|\nabla u\right| = \sqrt{u_{,x}^2(x,y) + \quad u_{,y}^2(x,y) + \quad v_{,x}^2(x,y) + \quad v_{,y}^2(x,y)} \tag{9.15}$$

and

$$\left|\nabla u\right|^2 = u_{,x}^2(x,y) + \quad u_{,y}^2(x,y) + \quad v_{,x}^2(x,y) + \quad v_{,y}^2(x,y) \tag{9.16}$$

or

$$\left|\nabla u\right|^2 = \begin{Bmatrix} u_{,x} & u_{,y} & v_{,x} & v_{,y} \end{Bmatrix} \begin{Bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{Bmatrix} \tag{9.17}$$

To connect derivatives in Cartesian and parametric space the following relation is used

$$\begin{Bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{Bmatrix} = \frac{1}{\det[J]} \begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ -J_{21} & J_{11} & 0 & 0 \\ 0 & 0 & J_{22} & -J_{12} \\ 0 & 0 & -J_{21} & J_{11} \end{bmatrix} \begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{Bmatrix} \Leftrightarrow$$

$$\begin{Bmatrix} u_{,x} & u_{,y} & v_{,x} & v_{,y} \end{Bmatrix}^T = \left[\overline{J}^{-1}\right] \begin{Bmatrix} u_{,\xi} & u_{,\eta} & v_{,\xi} & v_{,\eta} \end{Bmatrix}^T \tag{9.18}$$

As a result

$$\left|\nabla u\right|^2 = \left\{u_{,\xi} \quad u_{,\eta} \quad v_{,\xi} \quad v_{,\eta}\right\}\left[\bar{J}^{-1}\right]^T\left[\bar{J}^{-1}\right]\begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{Bmatrix} \tag{9.19}$$

But

$$\begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{Bmatrix} = \begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & \cdots & N_{i,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & \cdots & N_{i,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & \cdots & 0 & N_{i,\xi} \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & \cdots & 0 & N_{i,\eta} \end{bmatrix}\begin{Bmatrix} d_1 \\ d_2 \\ \vdots \\ d_3 \end{Bmatrix} \Leftrightarrow$$

$$\left\{u_{,\xi} \quad u_{,\eta} \quad v_{,\xi} \quad v_{,\eta}\right\}^T = \left[\mathrm{B}_2\right]\{d\} \tag{9.20}$$

So finally

$$\left|\nabla u\right|^2 = \{d\}^T\left[B_2\right]^T\left[\bar{J}^{-1}\right]^T\left[\bar{J}^{-1}\right]\left[B_2\right]\{d\} \tag{9.21}$$

The expression of the estimated error per element takes now the following form

$$\varepsilon_e = \frac{1}{\det\left[J\right] t d\xi d\eta}\left(\int\int\{d\}^T\left[B_2\right]^T\left[\bar{J}^{-1}\right]^T\left[\bar{J}^{-1}\right]\left[B_2\right]\{d\}\det\left[J\right]d\xi d\eta\right)^{1/2} \tag{9.22}$$

Numerical integration procedures for ξ,η lead to integration of tensor product Gauss points.

$$\varepsilon_e = \frac{1}{\det\left[\mathrm{J}\right]\mathrm{t}\mathrm{d}\xi\mathrm{d}\eta}\sum_{i=1}^{GP\xi}\sum_{j=1}^{GP\eta}\{\mathrm{d}\}^T\left[\mathrm{B}_2\right]^T\left[\bar{\mathrm{J}}^{-1}\right]^T\left[\bar{\mathrm{J}}^{-1}\right]\left[\mathrm{B}_2\right]\{\mathrm{d}\}\det\left[\mathrm{J}\right]\mathrm{w}_i^{GP\xi}\cdot\mathrm{w}_j^{GP\eta} \tag{9.23}$$

Considering the previous equations the error per element can be calculated numerically. If ε_e is larger than the C times sum of errors

$$\varepsilon_e > C\sum_{1}^{n_e}\varepsilon_e \tag{9.24}$$

then the element has to be refined. The introduced constant C is in fact a percentage of the sum of errors. For example it could be 5%. Any element with error larger than 5% of the total error is selected for refinement. Parameter C can be adjusted according to the problem solved.

The value given to C can be handful to restrict the adaptive refinement expanding area by increasing its value. Increase of C can lead to fewer indicated elements. In first view it can be said that this would decrease the achieved precision. But it will be shown in applications that due to the expanded nature of adaptive refinement it can yield the same or equivalent precision even though the marked for refinement elements are fewer. The overlapping nature of basis functions tends now to turn itself into a valuable advantage.

In some cases where the gradient of the exact solution has a large gradient on a region, the adaptive refinement process may never end. This obstacle is overcome by comparing the global gradients of two consecutive levels. If the two gradients have a difference small enough, the global domain will not be refined anymore. Global error is defined by the following equation:

$$\varepsilon_g = \frac{1}{V_g}\left(\int_{\Omega_g} |\nabla u|^2 \, d\Omega_g \right)^{1/2} \tag{9.25}$$

where Ω_g is the global domain and V_g the corresponding volume.

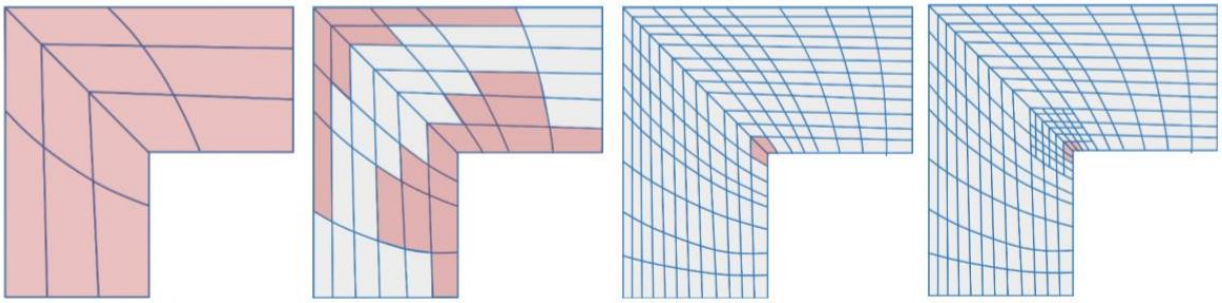


Figure 9.7.
Elements indicated by error estimator to be refined in adaptive hierarchical refinement sequence of levels. (Iakovos Antonios 2015)

9.3 VALIDATION OF ADAPTIVE HIERARCHICAL REFINEMENT

Having an adaptive algorithm now in hand, it is time to define some ways to validate its produced results. In other words it is very important to be able to compare its efficiency with equivalent methods that are already established and used.

The most common refinement technique that can be used to make a valid comparison is uniform refinement. This comparison will be also adopted by the present thesis. For this purpose two new names describing uniform refinement are introduced, to facilitate its differentiation according to some modifications used in the way they are applied. These two names are ‘’uniform refinement’’ and ‘’adaptive uniform refinement’’. Their difference will be explained in the following paragraphs.

‘’Adaptive uniform refinement’’ follows the same procedure as hierarchical adaptive refinement. The initial parameterization of the problem is solved, elements with high error are indicated by the error estimator and then are refined by inserting new knots in the parametric knot value vectors. This procedure has an iterative nature until there are no more elements surpassing the predefined error tolerance.

The method named simply ‘’uniform refinement’’ and is applied in this thesis depends on the results of the adaptive hierarchical refinement. Adaptive hierarchical is first applied on the parameter domain of the problem. Knots are then inserted in the parametric knot value vectors using the h-refinement method in order to replicate the local mesh created in the area where adaptive refinement is expanded. The final purpose is to compare the local nature of each method and the influence it has on the solution’s precision.

It may sound inaccurate to compare adaptive hierarchical refinement with uniform refinement as it was introduced previously in this thesis, as their application is not independent. Adaptive uniform refinement on the other hand sounds as the only legit way to validate the method’s efficiency. In the frame of the actual results this is true. But comparing the method with just uniform refinement can be proved less inaccurate than it can be considered.

As it will be later presented in the unit of applications it is very important to reduce the area where adaptive refinement is expanded. It may seem more local than uniform refinement, as it is a method not characterized by the full tensor product property which leads to the introduction of unnecessary control points increasing the number of degrees of freedom. But in the case of adaptive refinement full tensor property is sort of replaced by the interaction of elements. Shape functions expand in more than one element increasing their influence in the approximation space. So, it will be shown that in the case of adaptive refinement the same accuracy can be achieved with fewer degrees of freedom, by refining basis functions whose support is closer to the area of high errors. In this sense it would be more efficient to suggest that the adaptive influence area could coincide with the area of high error that is influenced by adaptive uniform refinement.

Another fact that suggests that comparison with uniform refinement can be also considered efficient is the scale of problems. In large scale problems adaptive influence areas are too small in comparison to those corresponding to uniform refinement due to the full tensor product property. So it is legit to say that the in the area of interest the inserted degrees of freedom by the two techniques are pretty much the same. This can be assumed as higher errors will not be detected in one or two elements but in a number of elements that that describe an area. So the expansion of hierarchical shape functions will be considered less in comparison to the case of small scale problems where the number of necessary and not necessary basis functions can be comparable.

So, to express the relation between the number of hierarchical and uniform B-Splines in some pretty small problems it was chosen to introduce the comparison between adaptive and uniform refinement in addition to the comparison with adaptive uniform refinement. To clarify all the aforementioned aspects some mesh examples are presented in **Fig. 9.8**.

Now it is time to validate the efficiency of adaptive hierarchical refinement. The first examined aspect is convergence tolerance. As it was explained in the unit about error estimator a constant C is defined so as to check which elements will be indicated for refinement. This constant C leads to all the elements with error higher than C % of the total error. So it is vital to apply some parametric analyses in order to predefine this constant. It

has to be said that constant C is an empirically fine-tune parameter for the specific problem. By using several convergence tolerances it is easy to define the degrees of freedom-error relations in order to choose the convergence that yields the same or almost the same precision with all the other cases but with less degrees of freedom.

The second comparison refers to the relation of degrees of freedom and errors derived from the three methods that were previously described and explained. This can be considered the most important comparison of all as it indicates which method yields better results with less degrees of freedom.

Another aspect that can be examined to validate adaptive hierarchical refinement is the form of stiffness matrices, their sparsity and bandwidth. Condition numbers could also be examined but are not taken into account in the present thesis. In this sense it is also very important to examine how stiffness matrices can be factorized. In this thesis the presented factorization is LU.

Now that the validation tools are described, they will be applied in the applications analyzed in **Chapter 10**.

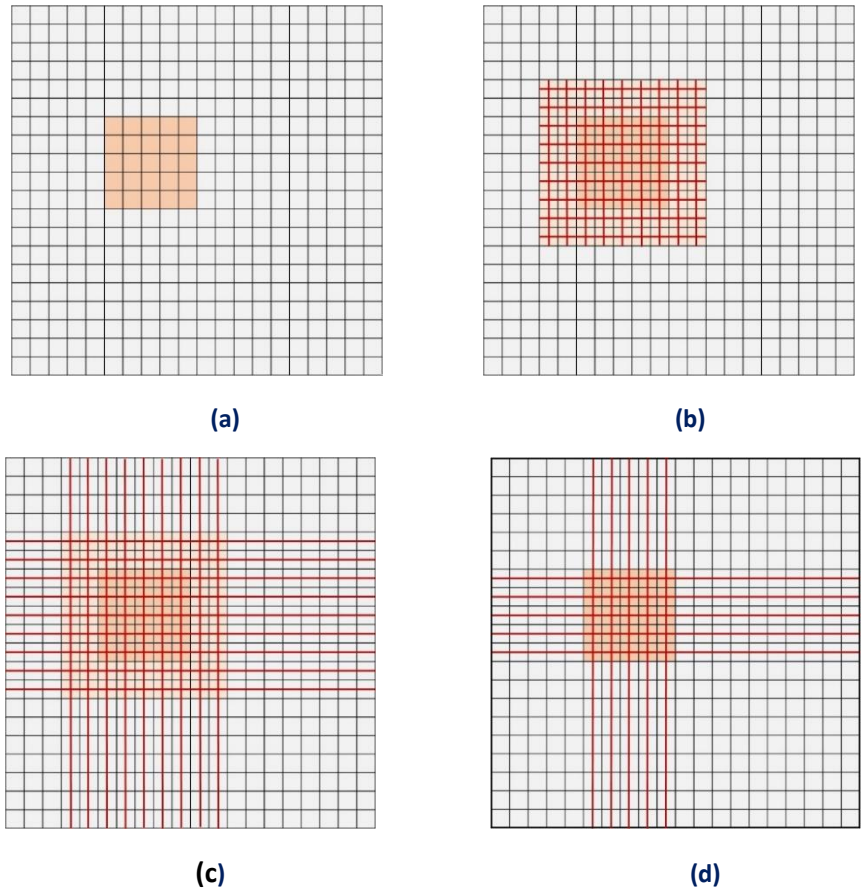


Figure 9.7.
(a) Area marked for refinement. Meshes for **(b)** Adaptive Hierarchical , **(c)** Uniform and **(d)** Adaptive uniform refinement.
(Iakovos Antonios 2015)

10 . APPLICATIONS

10.1 INTRODUCTION

In this chapter will be presented the application results that derived from the adaptive refinement algorithm proposed in **Chapter 9**. Two examples were chosen to validate the adaptive hierarchical refinement method. The first one is a 2D cantilever to feature the use of adaptive refinement in a case of concentrated loads. The second application concerns an L-Shaped domain to feature the application of adaptive refinement in areas where change of geometry introduces concentrated stresses. Both examples are subjected to plane stress static isogeometric analysis. Their results proved to be very useful in understanding how adaptive mechanism works in isogeometric analysis and led to some interesting conclusions able to help improve the existing procedure.

10.2 CANTILEVER 2D

The first application is a 2D cantilever. The cantilever presented in **Fig. 10.1** is subjected to plane stress. The dimensions of the cantilever are 1 m length and 1 m height. Its third dimension (thickness) is significantly smaller than the other two (0.1 m). The degrees of freedom on the left side are all fixed. A concentrated load P=200 kN is applied on the upper right corner of the cantilever. The purpose of this problem is to feature the use of adaptive refinement in cases where local refinement is needed. In this case it is expected that error will be significant higher in the area around the concentrated load where stress concentration will appear.

The initial parametrization of the cantilever consists of quadratic basis functions on axes ξ, η. The initial parameter domain is defined by two parametric knot value vectors Ξ=[0 0 0 0.5 1 1 1] and H=[0 0 0 0.5 1 1 1]. Thus there are nine control points whose weights are considered to be equal to 1. So the initial parametrization consists of four knot span elements as can be seen in the following figure.

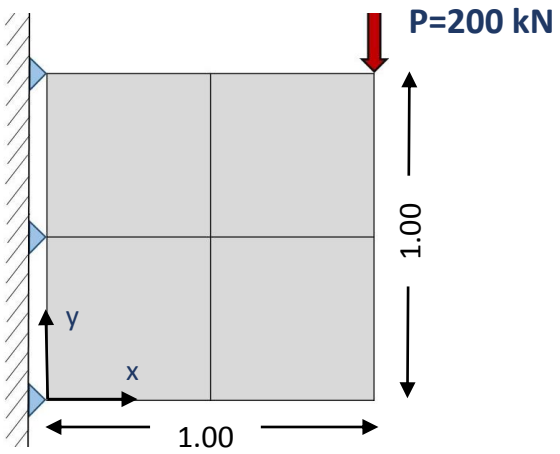


Figure 10.1.
Initial parametrization of the 2D cantilever subjected to plane stress.
(Iakovos Antonios 2015)

The present problem will be analyzed with the three different techniques that where introduced and analyzed in **Chapter 9**. These techniques were adaptive hierarchical refinement, uniform refinement and adaptive uniform refinement.

Starting with adaptive hierarchical refinement it was vital to examine how the convergence constant C used for the error indicator affects the precision of the analysis. Let's remind how constant C is used.

$$\varepsilon_e > C\sum_{1}^{n_e}\varepsilon_e$$

Constant C expresses in fact what percent of the total error an element has. If this percent is greater than the one chosen the element is selected for refinement. So it was decided to apply adaptive refinement for different C constant values to see how problem's precision responds. The values of constant C chosen were 2%, 3%, 5% and 7%. The relation of degrees of freedom and error expressed by the L_2 norm for different C values is presented in **Fig. 10.2**.

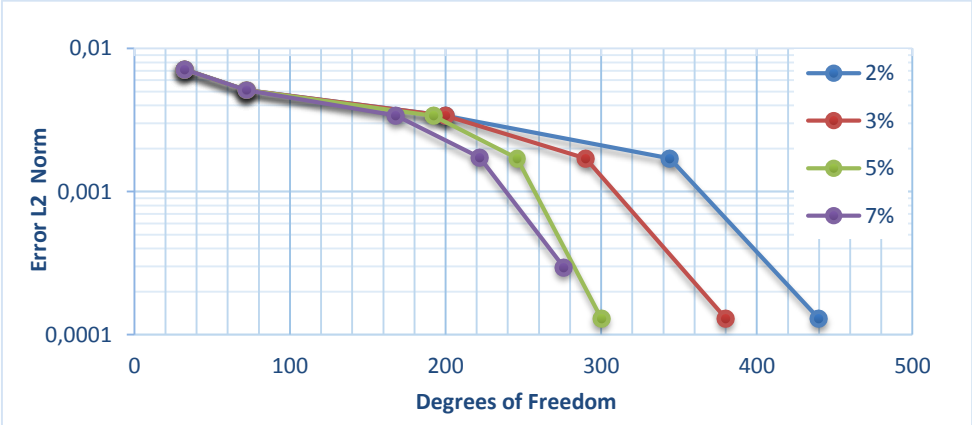


Figure 10.1.
Error in L_2 norm for different values of convergence constant C
(Iakovos Antonios 2015)

One should suggest that increasing constant C would increase the precision of the solution. That is not always the case in adaptive hierarchical refinement in isogeometric refinement. As shown in **Fig. 10.2** precision is not always sensitive to the increase of C. This can be explained considering the nature of hierarchical refinement. When an element is chosen for refinement all shape functions interacting with this element are replaced by finer shape functions. As shape functions in isogeometric analysis have a support of multiple knotspan elements, the refinement of one element affects the corresponding neighbor elements that shared the same shape functions. So the solution is enhanced in more than one element marked or not for refinement by the error indicator. **Fig. 10.3** illustrates this case to better understand how the described mechanism works.

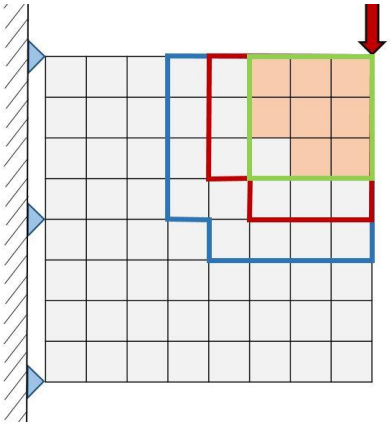


Figure 10.3.
Subdivided elements for different values of convergence constant C
(Iakovos Antonios 2015)

In the above figure the shaded area depicts the elements that were indicated for refinement for C=2 % in the second level of adaptive refinement. The colored lines present the union of elements that were subdivided for convergence constants 2% (blue), 3% (red) and 5 % (green) respectively. Even though the area of influence decreases with the increase of C the precision remains almost the same. This is justified by the previous figure representation because the support of shape functions that were refined is containing the marked element area of convergence 2% in all cases. That is a very important aspect as it suggests that adaptive refinement can be more localized with only restriction that the refinement area always contains the area of higher errors. The expansion to neighbor elements seems no to participate in the improvement of solution. That is a very promising conclusion for the efficiency of adaptive refinement, as error could be decreased with even less degrees of freedom.

Now that convergence constant has been examined it is time to move forward to other useful comparisons. In order to proceed it was decided to examine the case of convergence constant C= 5 % which yields better results with fewer degrees of freedom. The same convergence constant C will be later used in adaptive uniform refinement.

Five levels of adaptive refinement were applied on the present application. As it was expected and shown in previous figures adaptive refinement was concentrated in the area

of the concentrated load. The two first levels coincide with the case of global uniform refinement as all the elements were indicated for refinement. The sequence of indicated elements and adaptive refinement levels is shown in **Fig. 10.4**.

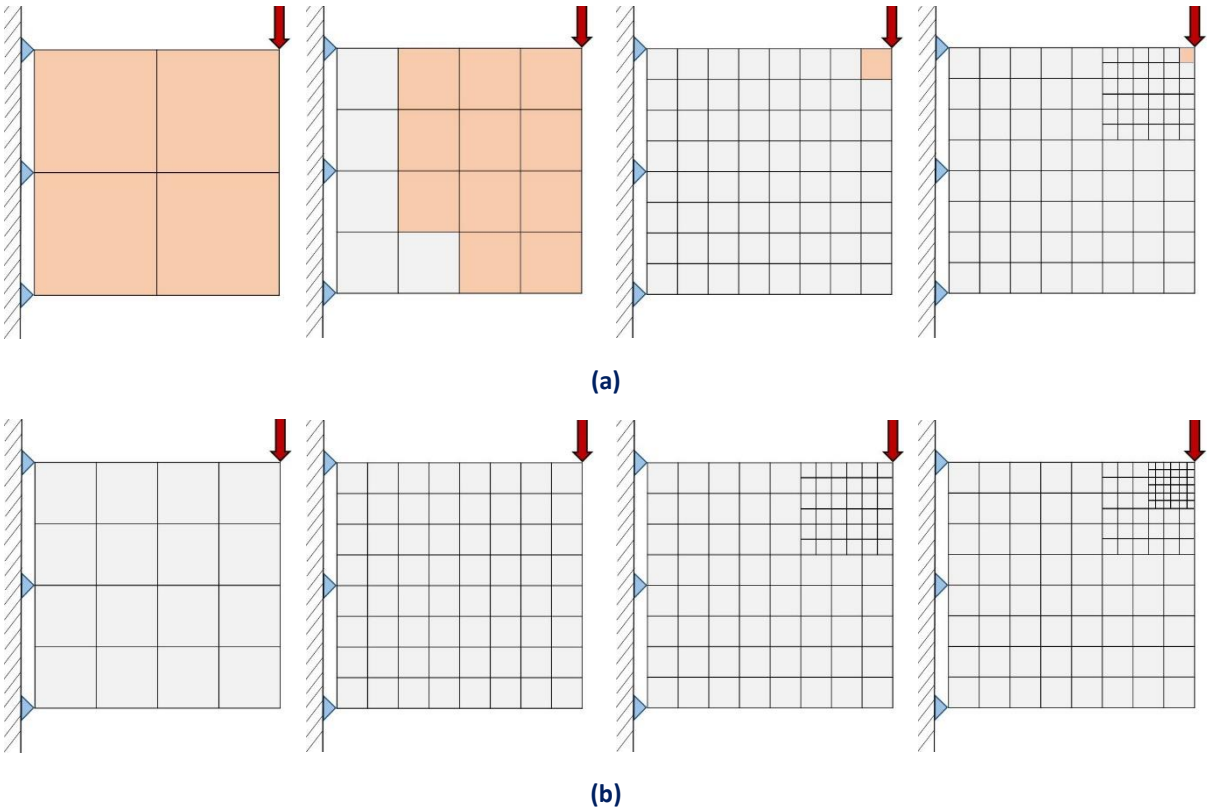


Figure 10.4.
(a) Sequence of elements indicated for refinement in the four first levels **(b)** Sequence of adaptive meshes produced for levels 2-5 in respect to the elements marked for refinement in previous levels as shown in the first row of the figure.
(Iakovos Antonios 2015)

Adaptive refinement seems to improve the precision of the analysis as shown in **Fig. 10.5**. The diagram depicts the decrease of error, expressed with the L_2 norm, in every level of the adaptive hierarchical refinement procedure.

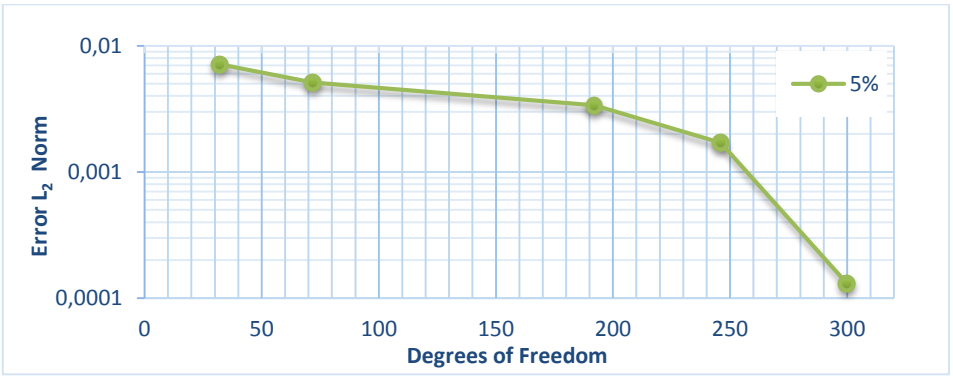


Figure 10.5
Error in L_2 norm for convergence constant C= 5%
(Iakovos Antonios 2015)

Some of the displacements and stress contours are presented in the following figure. It is obvious that in the area of the upper right corner the influence of the concentrated load is minimized both for displacement and stresses. The contours depict corresponding fields in the first and fifth level of adaptive refinement

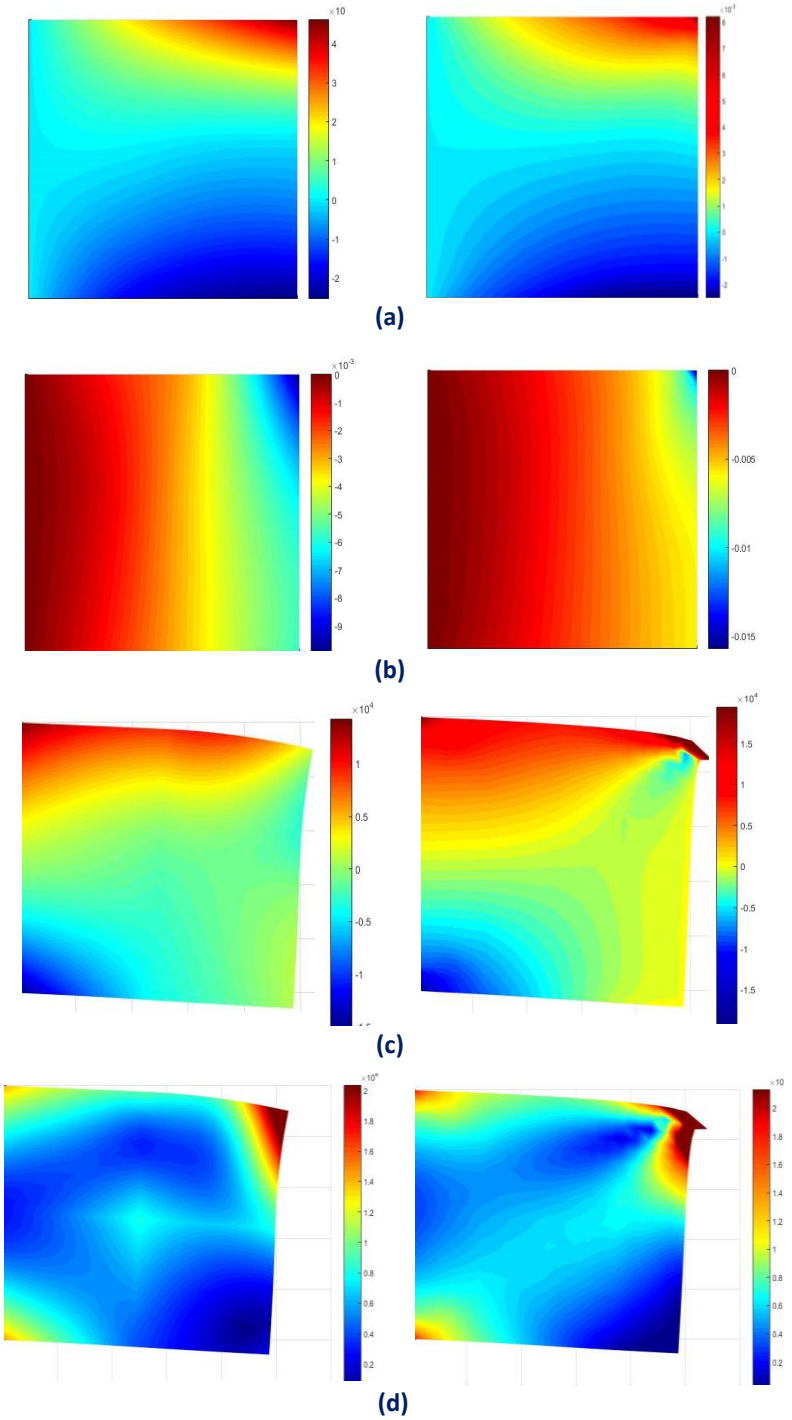


Figure 10.6.

(a) Displacement X, **(b)** Displacement Y, **(c)** Stress σ_{xx}, and **(d)** Stress Von Mises of the deformed cantilever in first and fifth level respectively
(Iakovos Antonios 2015)

Adaptive refinement is now compared with uniform and adaptive uniform refinement. Adaptive uniform refinement is executed with the same convergence constant, meaning C=5 %. Uniform refinement as explained is applied as explained in **Chapter 9.** Knots were inserted in knot value vectors ξ and η respectively to replicate the mesh in the area of interest. The corresponding final meshes (level 5) meshes produced with the three aforementioned methods are presented in **Fig. 10.7**.

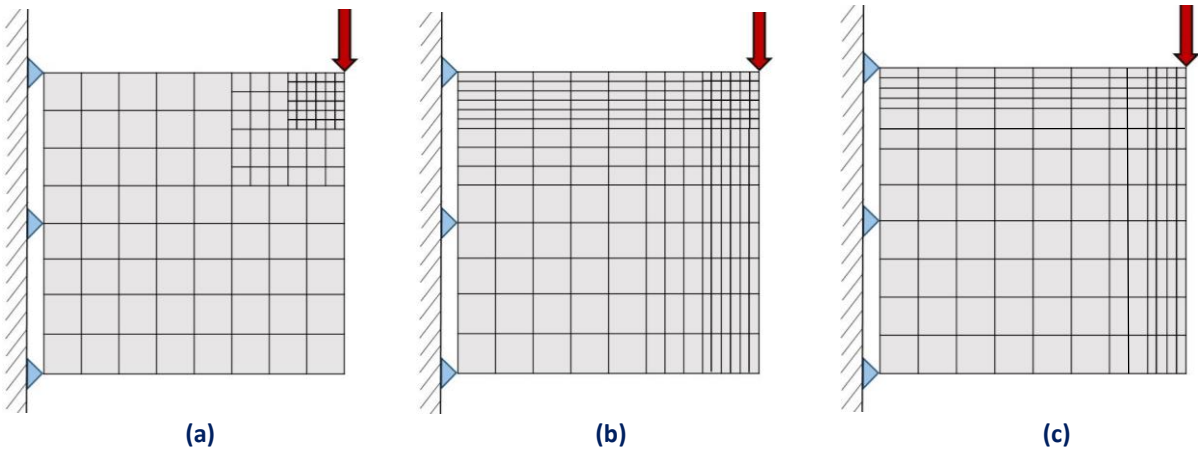


Figure 10.7.
(a) Adaptive hierarchical refinement mesh, **(b)** Uniform refinement mesh , **(c)** Adaptive uniform refinement mesh
(Iakovos Antonios 2015)

From the analysis results shown in **Fig. 10.8**. it seems that adaptive refinement is more efficient than the other two methods. It can yield same precision solution with less degree of freedom. It is more obvious when it is compared with uniform refinement for the reasons explained in the previous chapter. In level 5 the same accuracy can be achieved with 40 % fewer degrees of freedom. Taking into account that this difference will grow from level to level because of the increased number of inserted degrees of freedom due to the full tensor product property of uniform refinement. It is positive that adaptive refinement is even more efficient than adaptive uniform refinement.

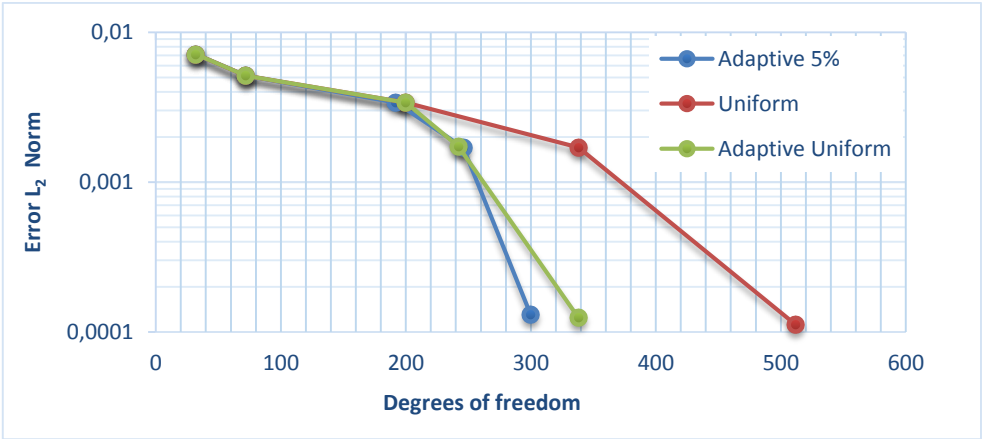


Figure 10.8.
Comparison between adaptive (blue), uniform (red) and adaptive uniform (green) refinement.

But precision in adaptive refinement has another cost that has to be taken into account for the validation of the proposed method. Even though degrees of freedom-error relation counts as positive for adaptive refinement it is still a method more cost intensive in the formulation of stiffness matrix. This could not always count as a drawback as the reducing of computational cost for the solution could affect the needed total cost. For this reason it is very important to take into consideration the form of stiffness matrices. It will be shown that adaptive refinement leads to sparse matrices with variable bandwidth.

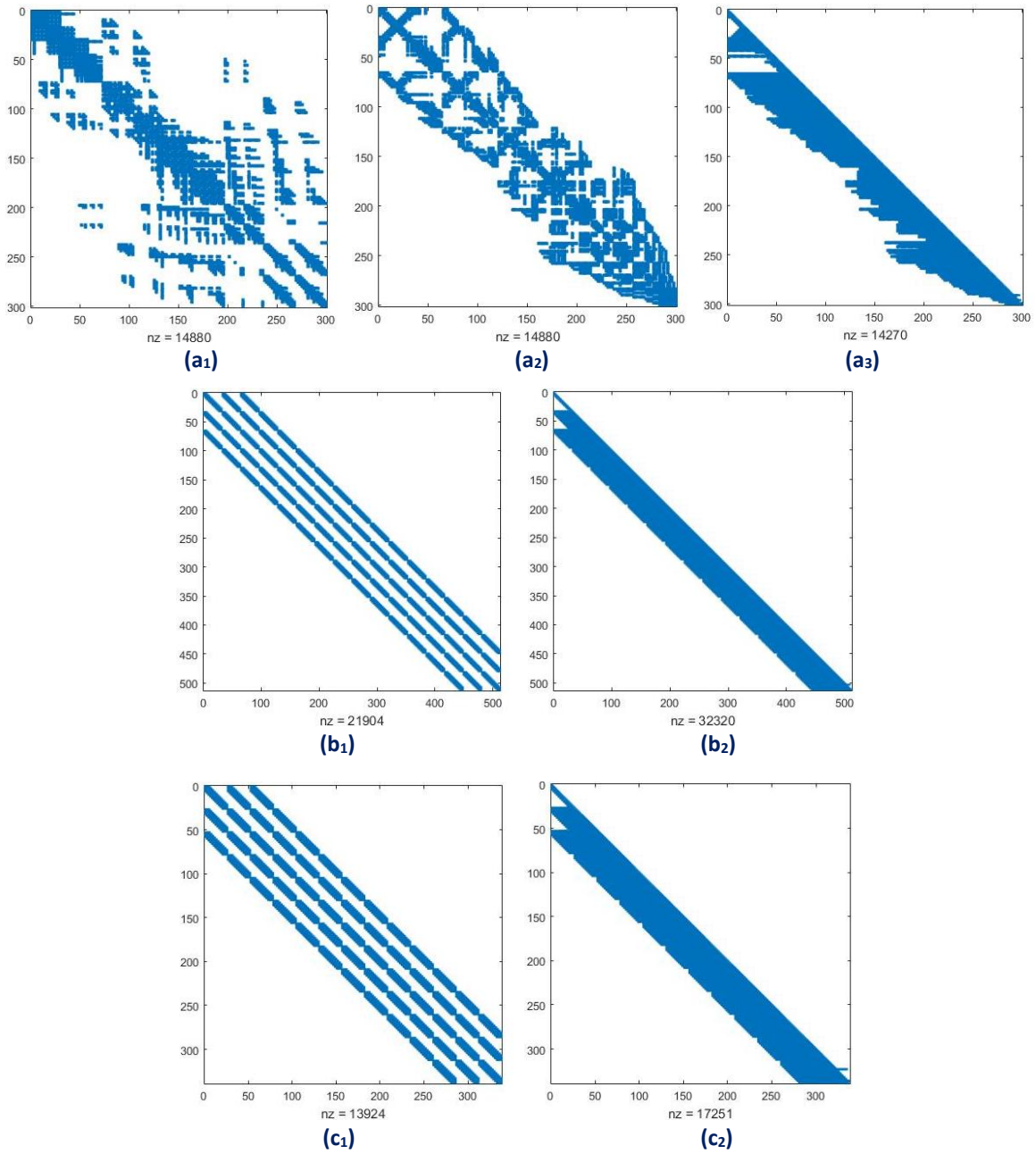


Figure 10.9.
Level 5 stiffness matrices
(a) Adaptive hierarchical refinement.1)Stiffness matrix. 2) Stiffness matrix rearranged with Cathill-McKee Algorithm. 3) Lower LU factorization matrix **(b)** Uniform refinement. 1) Stiffness matrix. 2) Lower LU factorization matrix **(c)** Adaptive uniform refinement. 1) Stiffness matrix. 2) Lower LU factorization matrix

10.3 L-SHAPE 2D

The second example is a 2D L-Shape subjected to plane stress isogeometric analysis. The L-Shape is shown in **Fig. 10.10**.

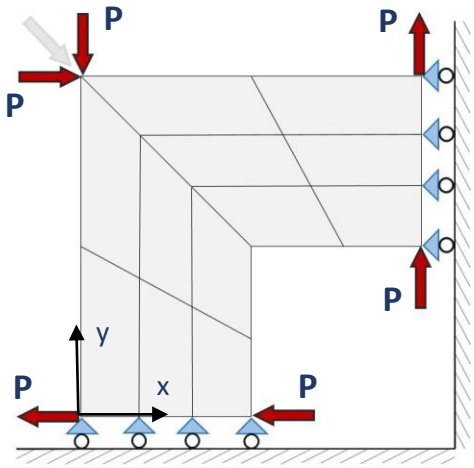


Figure 10.10.
Initial parametrization of the 2D cantilever subjected to plane stress.
(Iakovos Antonios 2015)

Let's now describe its geometry. The long sides of the L-Shape are 1 m long and the short ones 0.5 m respectively. The thickness is considered small enough in comparison to the other dimensions and is defined to be 0.1 m. The lower side of the L-Shape is fixed along direction y and the right side fixed along axis x. All the loads are applied on interpolatory boundary control points as shown in the previous figure. The value of all applied loads is P=150 kN. This problem was chosen to feature the need of adaptive refinement not only on areas where concentrated loads are applied but also in areas where the change of geometry leads to concentration of stresses.

The initial parametrization of the L-shape consists of quadratic basis functions on axes ξ, η. The initial parameter domain is defined by two parametric knot value vectors Ξ=[0 0 0 1/3 2/3 1 1 1] and H=[0 0 0 1/4 2/4 3/4 1 1 1]. Two pairs of double control points were used to create the bending corners of the L-Shape. Thus there are thirty control points whose weights are considered to be equal to 1. So the initial parametrization consists of twelve knot span elements as can be seen in **Fig. 10.10**.

The steps of evaluation of adaptive refinement in the case of L-shape are the same used for the cantilever 2D problem. First it is important to choose a convergence constant C for the error indicator. It has to be clarified that this process is only applied in order to validate the sensitivity of error to the constant C. In case one problem has to be solved it is obvious that this constant C is defined empirically without applying parametric investigations. The chosen constants are 2%, 3%, 4% and 5%. The results are represented in **Fig. 10.11**. It seems that the best results considering the relation degrees of freedom-error are yielded with the use of convergence constant C=4 %. For C=5 % error seems to stabilize after the second iteration. This happens because a high error area is missed during refinement selection. The

result is that the corresponding error to this area is never decreased , cause local shape functions are not refined. The convergence constant used to proceed to the next comparisons is finally 3 % as C= 4% might behave exactly as 5 % if another iteration is executed. C=4 % marks only the area around the inside corner.

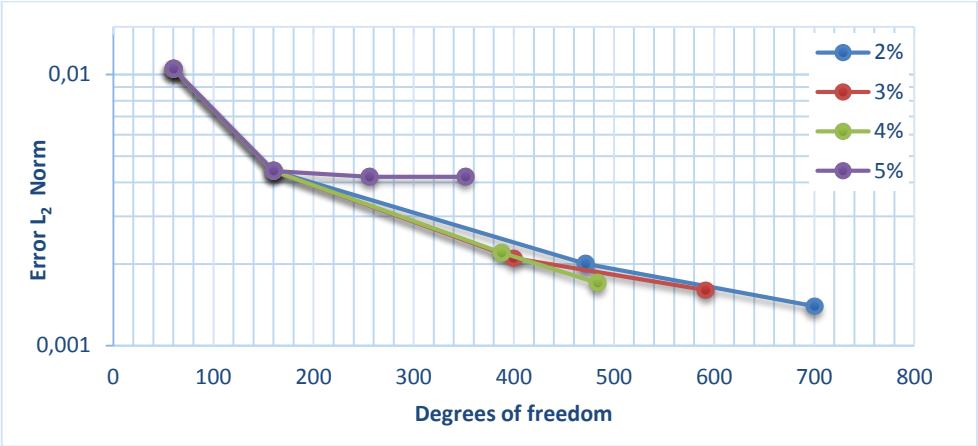


Figure 10.11.
Error in L_2 norm for different values of convergence constant C
(Iakovos Antonios 2015)

Four levels of adaptive refinement were applied on the present application. Error indicator marked the elements that were expected according to **Fig. 10.12**.

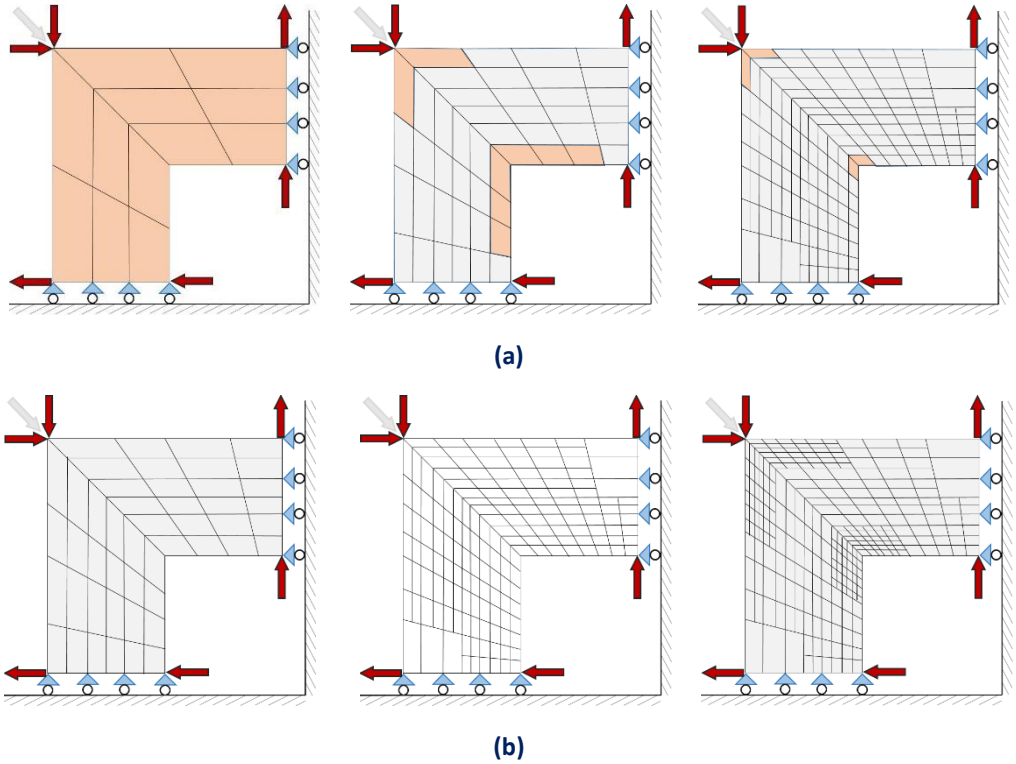


Figure 10.12.
(a) Sequence of elements indicated for refinement in the three first levels **(b)** Sequence of adaptive meshes produced for levels 2-4 in respect to the elements marked for refinement in previous levels as shown in the first row of the figure.
(Iakovos Antonios 2015)

As it was expected the error estimator indicated elements that were near concentrated loads or the inside corner of the L-shape where stress are concentrated. Some of the results after four level of adaptive refinement are shown in **Fig. 10.13**.

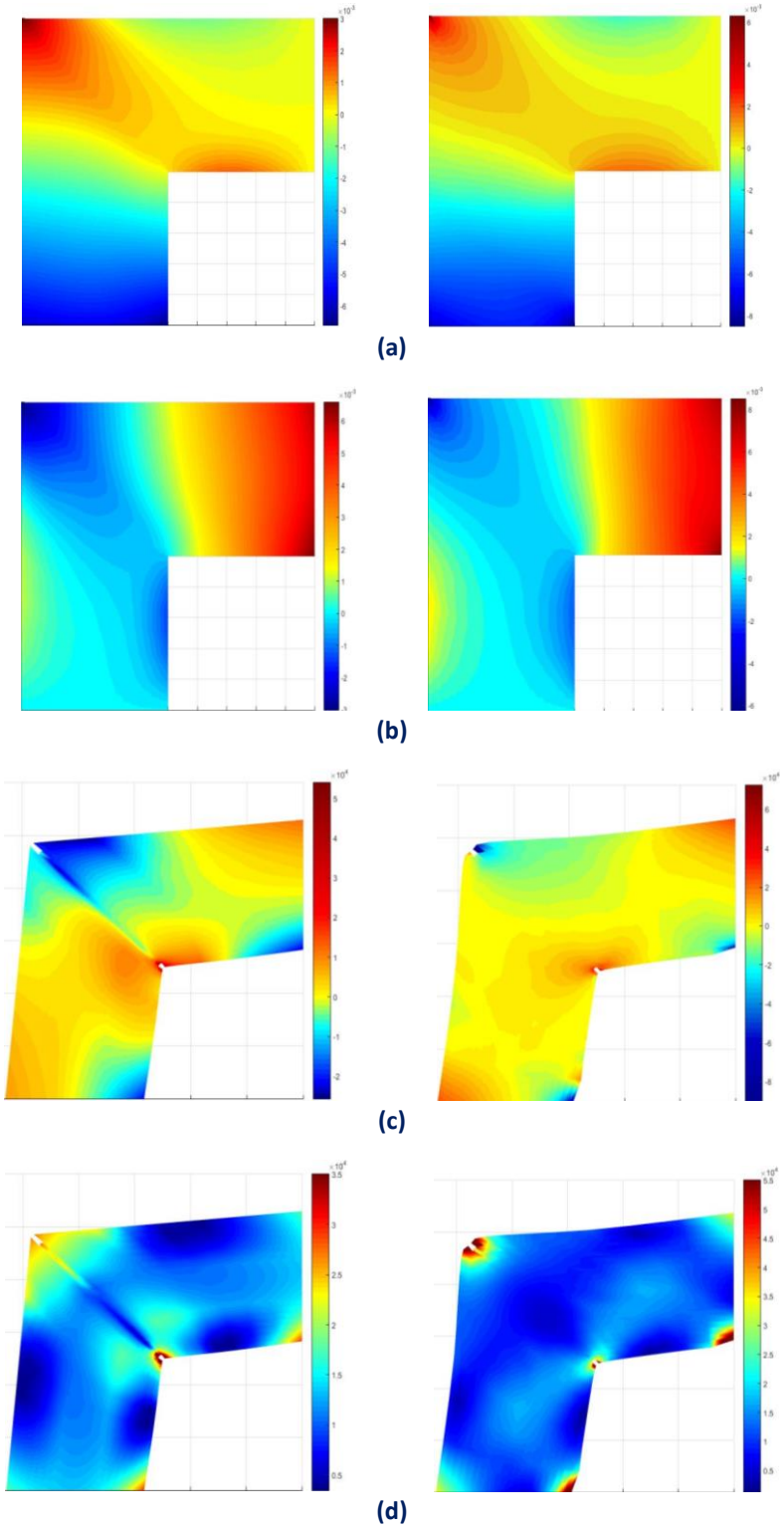


Figure 10.13.
(a) Displacement X, **(b)** Displacement Y,**(c)** Stress σ_{xx}, and **(d)** Stress Von Mises of the deformed cantilever in first and fifth level respectively
(Iakovos Antonios 2015)

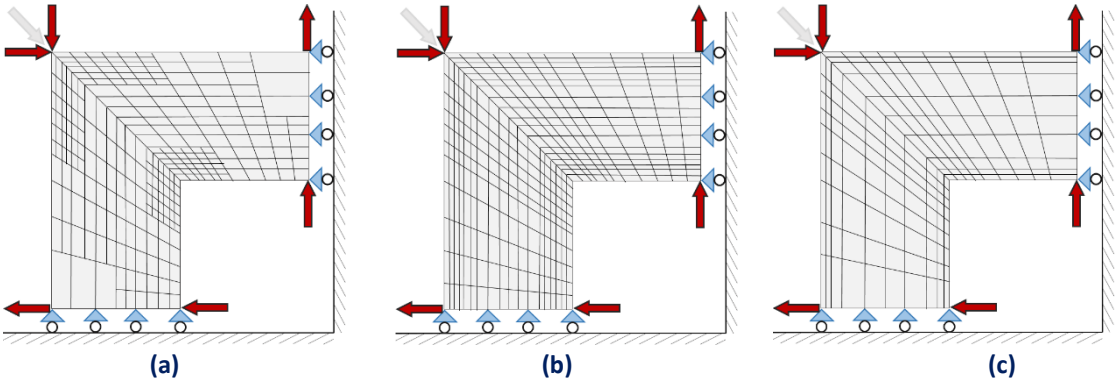


Figure 10.14.
(a) Adaptive hierarchical refinement mesh, **(b)** Uniform refinement mesh , **(c)** Adaptive uniform refinement mesh
(Iakovos Antonios 2015)

It is now time to compare adaptive hierarchical refinement with uniform and adaptive uniform refinement. The convergence constant used for adaptive uniform refinement is still 3%. The result meshes of each method are presented in **Fig. 10.14**.

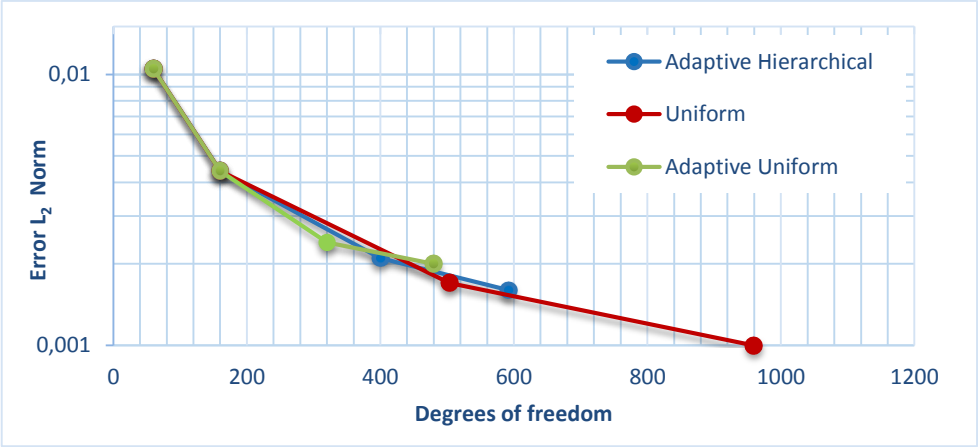


Figure 10.15.
Comparison between adaptive (blue), uniform (red) and adaptive uniform (green) refinement.

The analysis results produced by the three different methods are presented in **Fig. 10.15.** In this case it is not clear which of the three methods is more efficient. The degrees of freedom-error curves tend to coincide. This result can be easily explained. The error indicator marked only some of the areas of higher level. Areas near some concentrated loads where not indicated for refinement. It would be necessary to decrease the convergence constant C. But this choice would also lead to similar results. In this case the refined areas would cover large part of the geometry and the main advantage of adaptive refinement, which is local nature, would be automatically canceled. If the convergence constant was decreased to indicate more elements for refinement, results as those presented for the cantilever, would be possible after a larger number of iterations. The insertion of unnecessary control points due to the full tensor product property of B-Splines in combination with the decrease of error by the adaptive refinement applied in all the areas of interest would lead to results more positive for the adaptive procedure. It has to be

said that even in the examined case, adaptive refinement tends at least to yield results similar to the ones corresponding to uniform refinement and adaptive uniform refinement.

Finally it is also important to compare the stiffness matrix properties for each case of refinement. Yielding the same precision results doesn't mean that all methods are equivalent. Their matrices condition, sparsity, bandwidth and of course the needed cost to formulate them have to be considered. Adaptive refinement for example leads to sparser matrices with variable bandwidth as was also noticed in the previous example (cantilever 2D). This kind of matrices are not always easy to handle in comparison to the matrices produced by the other two types of refinement. They are worth for analysis only when they can lead to the accuracy with far less degrees of freedom so as to balance all the other disadvantages they have.

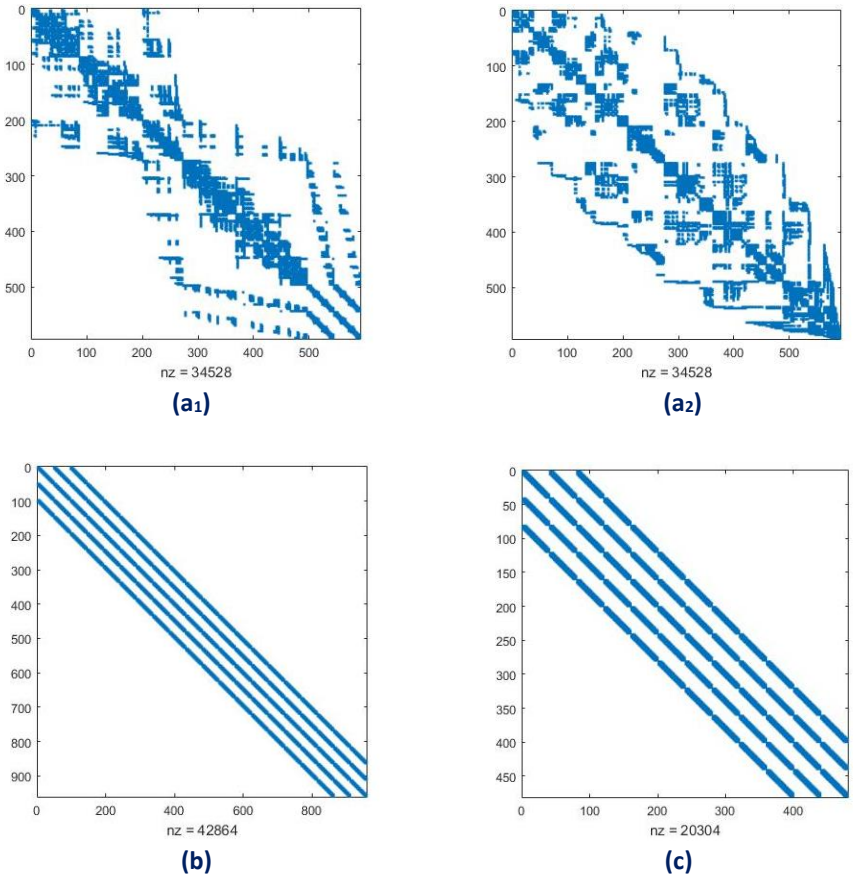


Figure 10.9.
Level 5 stiffness matrices **(a)** Adaptive hierarchical refinement.1)Stiffness matrix. 2) Stiffness matrix rearranged with Cathill-McKee Algorithm **(b)** Uniform refinement. Stiffness matrix. **(c)** Adaptive uniform refinement. Stiffness matrix.

11 . CONCLUSIONS AND PROPOSALS

11.1 GENERAL CONCLUSIONS ON ADAPTIVE HIERARCHICAL REFINEMENT

It is well known that a major drawback of isogeometric analysis was until recently the lack of a local refinement mechanism. Knot insertion or order elevation alter larger areas of the approximate space, increasing dramatically the number of degrees of freedom. Due to the tensor product nature of B-Splines it was impossible to locally refine an area of high error without introducing unnecessary shape functions. For h-refinement this not only increased the degrees of freedom, but it also created bad size ratio elements leading to instabilities in numerical analysis.

The aforementioned aspect in combination with the research effort to enrich isogeometric analysis with procedures already known from FEM made the need of introducing a local refinement technique obligatory in order to make isogeometric analysis a competitive analysis tool.

In this sense researchers introduced hierarchical refinement. Hierarchical refinement, refers to the hierarchy of the refined levels which are characterized by their nested relation. All its variations seem to be promising in order to boost not only the accuracy of the solution but also the CPU time consumption. Even though the formulation of the basis and the stiffness matrix is a more complex procedure than the corresponding ones used in other classical refinement methods it can become competitive by yielding same precision results with far less degrees of freedom. Hierarchical refinement, practically replaces coarse grid basis functions, with finer ones in order to achieve better local solution. From the initial definition given by Vuong et al, even though the local refinement in IGA has been accomplished, several side effects have been presented such as:

- Linear independence of the hierarchical basis.
- Partition of unity property across the hierarchical basis.
- Evaluation procedure of proper weights in order to restore the partition of unity.
- Formulation of new global stiffness matrix in each level of refinement.
- Expansion of the refinement area due to the multi-element support of shape functions.

Bornemman and Cirak tried to cope with some of the aforementioned issues by introducing a more algorithmic methodology. They introduced a straightforward way to evaluate shape

functions scale factors in order to satisfy the partition of unity property. These scale factors were defined as proposed by Kraft in 1997, applying the subdivision property used in refinement of CAD geometries. The succeeded to simplify it even more by corresponding finally the scale factors to the control points and not the shape functions.

But even though the proposed methodologies yielded satisfying results they were characterized by a very important drawback. Due to the numerous interactions between the shape functions that remained intact or were inserted during a hierarchical level of refinement, stiffness matrices' patterns seemed not so positive. Stiffness matrices were characterized by increased sparsity and variable bandwidth, properties that increase computational cost.

To overcome this obstacle, Giannelli introduced truncated B-Splines. The truncation mechanism managed to reduce the overlapping between the coarser functions and the finer ones. This has been translated in significantly reduced sparsity of stiffness matrices. The most important feature of truncation, was that the basis did not need anymore the subdivision weights in order to satisfy partition of unity and thus, the formulation of the stiffness matrix was straightforward.

Even though, truncated B-Splines managed to deal with the partition of unity property, they did not manage to narrow the refined area and thus achieve a more local nature of hierarchical refinement. D. Schillinger tried to produce meshes where the refinement area were more local than previous suggested methods. Instead of subdividing shape functions that were indicated for refinement, he imposed a set of finer shape functions in the area of interest avoiding the expansion to collateral elements. Then the truncation mechanism was used to satisfy the properties of linear independence and partition of unity. The combination of his localization idea with the truncation methodology led to further decrease of degrees of freedom number and stiffness matrix sparsity.

The conclusions and the methodologies presented till now were based on the formulation of the hierarchical basis or stiffness matrix consequently. But it is very important to use the positive aspects of hierarchical refinement to further reduce the needed computational cost beyond the formulation of hierarchical bases by improving the solution procedures for such applications. In this frame the present thesis described the local solution method introduced by Wu et al. This method proposes an efficient way to decouple each hierarchical level's equations in order to avoid solving the whole problem all over again in every iteration step of adaptive refinement. Time consumption for the solution of the classical hierarchical refinement 1D, 2D and 3D cases stiffness matrices is accelerated dramatically, where at the same time optimum convergence is achieved.

Summarizing all the aforementioned methodologies the following conclusions can be derived:

- Same accuracy can be achieved with less degrees of freedom.
- Hierarchical refinement expands to collateral elements due to shape function overlapping, losing its local nature.
- Sparsity of stiffness matrices depends on the used refinement strategy. In any case it yields sparser matrices than those derived from uniform refinement.
- The total CPU time is immediately affected by the evaluation or truncation technique used in each strategy.
- The Wu et al proposal for solving each level independently, seems to be very promising in order to accelerate significantly local refinement in isogeometric analysis.

11.2 GENERAL CONCLUSIONS ON APPLICATIONS

Some interesting conclusions have also emerged from the applications presented in **Chapter 10**. The examples of cantilever and L-shape 2D were subjected to plane stress analysis. The analysis results were then compared with uniform and adaptive uniform methodologies as explained in **Chapter 9**.

Firstly it became apparent that error estimator plays a crucial role in adaptive hierarchical refinement. Constant C, which defines how strict or not is the selection of elements to be refined, is an important parameter that can affect the solution. Constant C has to be chosen very carefully in order to efficiently apply adaptive refinement. In the case of the cantilever the increase of constant C didn't affect the precision of the solution. Of course there is always a bound for its increase. It was observed that in the cantilever application greater values of C always defined shape functions whose support overlapped the area of higher error. So the same precision could be provided with the selection of the smallest number of shape functions interacting and overlapping with the area of interest.

The aforementioned observation is in fact the second conclusion, according to which adaptive refinement has possibilities to retain its locality by suitable selection of shape functions. The selection of all shapes functions that interact with elements that surpass the accepted error may introduce unnecessary shape functions and thus degrees of freedom. Adaptive refinement localization can thus be further improved.

Another observation is that adaptive hierarchical refinement increases its efficiency as hierarchical levels increase. That can be easily explained considering its nested property and the full tensor product property characterizing other methods. As the number of levels increases the refinement area tends to become smaller to satisfy the nested property of hierarchical levels. On the other knot insertion continues to overrun all the elements that are defined by a certain parametric position introducing more and more degrees of freedom. As the area of high error is now restricted the unnecessary shape functions tend to be even more than the ones that lead to local improvement of the solution. Vice versa, in the first levels of

refinement, adaptive refinement's behavior can be similar to uniform or adaptive uniform refinement as the majority of shape functions is selected for refinement. This case was encountered in the example of L-Shape 2D.

Finally, another conclusion that is also presented during the theoretical validation of the presented methods in the previous unit, concerns the sparsity pattern of stiffness matrices. It has been shown in the two explained applications that stiffness matrices corresponding to adaptive refinement lead to sparsity patterns not desirable in analysis. So even though the degrees of freedom-error relation seems positive for adaptive refinement the patterns of stiffness matrices can be considered a serious drawback for the effort made to increase degrees of freedom and computational cost needed for the solution of equations consequently. It has to be mentioned that the truncation technique was not used in the presented algorithm.

11.3 PROPOSALS

Conclusions always give birth to new proposals. What is really positive is that adaptive refinement is a research area which has many possibilities of improvement. Conclusions indicate the methodology subjects whose modification could lead to better solution.

In this sense it is obvious that adaptive refinement should be modified to retain the local nature it was meant for. Applications showed that the selection of less shape functions for refinement could lead to equally precise solution. The purpose is to define the less number of shape functions whose support van be fully or almost be defined by the high error area. This could be an efficient way to retain locality, but still has to be validated by adding it in the present algorithm in order to be applied.

It is also proposed to combine all the positive aspects adaptive hierarchical methods can provide. More specifically, the final target is to keep all the good properties of each method and eventually build-up an adaptive algorithm that could effectively tackle the majority of applications.

So, to this end, it would be an interesting idea to combine truncated B-Splines with the local solution method. This could be a realistic target, as local solution method is based on the existence of some characteristic regions of hierarchical bases that can be easily found in every hierarchical refinement method used until now. This combination could possibly lead to further reduction of the needed computational cost as both degrees of freedom and sparsity of stiffness matrices would possibly decrease.

The work presented in this thesis will continue by examining how the truncation mechanism affects the coarse B-Splines and their corresponding degrees of freedom. This is essential in order to be able to decouple each level from the other. Some modifications for the existing

methods should be possibly necessary in order to succeed in the implementation of the proposed hierarchical refinement scheme.

Another future target is the use of collocation methods to improve the cost intensive nature of adaptive refinement. Of what is already known collocation methods could boost adaptive refinement's efficiency.

11.4 SUBJECTS FOR FURTHER RESEARCH

Of course the Hierarchical Refinement area cannot be contained in just one thesis, as there are numerous other things to examine. In order for the method to mature, some of the following subjects may need further research:

- Use of collocation methods in hierarchical refinement
- Patch interconnectivity after hierarchical refinement.
- Adjusting the external loads and the boundary conditions, after the hierarchical refinement procedure.
- Evaluation and introduction of more elaborate error estimators.
- Hierarchical formulation of stiffness matrix.
- Conversion between T-SPLines and HB-SPLines, in a more compact manner in order to compare the two methodologies.
- The generalization of every theory for B-Splines in NURBS.

REFERENCES

1. Cotrell J.A., Hughes Th.J.R, Bazilevs Y., Isogeometric Analysis: Toward integration of CAD and FEA. 2009. Wiley.
2. Vuong, A.-V., Adaptive Hierarchical Isogeometric Finite Element Methods. 2012: Springer Science & Business Media.
3. Le Méhauté, A., C. Rabut, and L.L. Schumaker, Surface fitting and multiresolution methods. 1997: Vanderbilt University Press.
4. Wang, P., et al., Adaptive isogeometric analysis using rational PHT-splines. Computer-Aided Design, 2011. 43(11): p. 1438-1448.
5. Verhoosel, C., et al., Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone. Computer Methods in Applied Mechanics and Engineering, 2015. 284: p. 138-164.
6. Piegl, L. and W. Tiller, The NURBS book. 2012: Springer Science & Business Media.
7. Wei, X., et al., Truncated hierarchical Catmull–Clark subdivision with local refinement. Computer Methods in Applied Mechanics and Engineering, 2015. 291: p. 1-20.
8. Metsis, P., N. Lantzounis, and M. Papadrakakis, A new hierarchical partition of unity formulation of EFG meshless methods. Computer Methods in Applied Mechanics and Engineering, 2015. 283: p. 782-805.
9. Schillinger, D., et al., Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. Computer Methods in Applied Mechanics and Engineering, 2013. 267: p. 170-232.
10. Bornemann, P. and F. Cirak, A subdivision-based implementation of the hierarchical b-spline finite element method. Computer Methods in Applied Mechanics and Engineering, 2013. 253: p. 584-598.
11. Kuru, G., et al., Goal-adaptive isogeometric analysis with hierarchical splines. Computer Methods in Applied Mechanics and Engineering, 2014. 270: p. 270-292.
12. Nguyen-Thanh, N., et al., Isogeometric analysis using polynomial splines over hierarchical T-meshes for two-dimensional elastic solids. Computer Methods in Applied Mechanics and Engineering, 2011. 200(21): p. 1892-1908.
13. Schillinger, D., et al., 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, 2012. 249: p. 116-150.
14. Vuong, A.-V., et al., A hierarchical approach to adaptive local refinement in isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 2011. 200(49): p. 3554-3567.
15. Giannelli, C., B. Jüttler, and H. Speleers, THB-splines: The truncated basis for hierarchical splines. Computer Aided Geometric Design, 2012. 29(7): p. 485-498.
16. Giannelli, C., B. Jüttler, and H. Speleers, Strongly stable bases for adaptively refined multilevel spline spaces. Advances in Computational Mathematics, 2014. 40(2): p. 459-490.

17. Mokriš, D., B. Jüttler, and C. Giannelli, On the completeness of hierarchical tensor-product B-splines. Journal of Computational and Applied Mathematics, 2014. 271: p. 53-70.
18. Kleiss, S.K., B. Jüttler, and W. Zulehner, Enhancing isogeometric analysis by a finite element-based local refinement strategy. Computer Methods in Applied Mechanics and Engineering, 2012. 213: p. 168-182.
19. Wu, Z.-j., et al., A local solution approach for adaptive hierarchical refinement in isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 2015. 283: p. 1467-1492.
20. Johannessen, K.A., T. Kvamsdal, and T. Dokken, Isogeometric analysis using LR B-splines. Computer Methods in Applied Mechanics and Engineering, 2014. 269: p. 471-514.
21. Kiss, G., C. Giannelli, and B. Jüttler, Algorithms and data structures for truncated hierarchical B–splines, in Mathematical Methods for Curves and Surfaces. 2014, Springer. p. 304-323.
22. Buffa, A. and C. Giannelli, Adaptive isogeometric methods with hierarchical splines: error estimator and convergence. arXiv preprint arXiv:1502.00565, 2015.
23. Thomas, D.C., et al., Bézier projection: a unified approach for local projection and quadrature-free refinement and coarsening of NURBS and T-splines with particular application to isogeometric design and analysis. Computer Methods in Applied Mechanics and Engineering, 2015. 284: p. 55-105.
24. Giannelli, C. and B. Jüttler, Bases and dimensions of bivariate hierarchical tensor-product splines. Journal of Computational and Applied Mathematics, 2013. 239: p. 162-178.
25. Kiss, G., et al., Adaptive CAD model (re-) construction with THB-splines. Graphical models, 2014. 76(5): p. 273-288.
26. Scott, M.A., D.C. Thomas, and E.J. Evans, Isogeometric spline forests. Computer Methods in Applied Mechanics and Engineering, 2014. 269: p. 222-264.
27. Wang, Y., J. Zheng, and H.S. Seah. Conversion between T-Splines and Hierarchical B-Splines. in Computer graphics and imaging. 2005.
28. Zander, N., et al., Multi-level hp-adaptivity: high-order mesh adaptivity without the difficulties of constraining hanging nodes. Computational Mechanics, 2015. 55(3): p. 499-517.
29. Johannessen, K.A., 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, 2015. 291: p. 64-101.
30. Mokriš, D. and B. Jüttler, TDHB-splines: The truncated decoupled basis of hierarchical tensor-product splines. Computer Aided Geometric Design, 2014. 31(7): p. 531-544.
31. Karatarakis A., Karakitsios P, Papadrakakis M, A GPU accelerated computation of the isogeometric analysis stiffness matrix.
32. Papadrakakis M. Analysis of Structures with the Finite Element Method
33. Stamatis A., Isogeometric Linear Static Analysis with NURBS, Diploma thesis 2013
34. Tsapetis D., Isogeometric Static Analysis with T-SPLines, Diploma thesis 2014
35. Tripakis M., Hierarchical Formulation of the Isogeometric Analysis Method for Structures, Diploma thesis 2015
36. Karras D., Hierarchical Refinement in Isogeometric Analysis with NURBS, Diploma thesis 2015