Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών
Τομέας Μαθηματικών

Επίδοση της Αποσύζευξης Bézier στην Ισογεωμετρική Ανάλυση

Μεταπτυχιακή Εργασία
του
Γιάννη Ραδίτσα

Επιβλέπων: Χριστόφορος Προβατίδης
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2020

Περίληψη

Αυτή η εργασία έχει σαν σκοπό να μας εισάγει στα βασικά θέματα της ισογεωμετρικής ανάλυσης τόσο θεωρητικα, όσο και υπολογιστικά. Η θεωρία στην οποία θα αναφερθούμε θα περιέχει ενότητες όπως οι B-splines, οι NURBS και οι T-splines που βασίζονται στις B-splines καθώς και η έννοια της αποσύζευξης Bézier. Η αποσύζευξη Bézier θα μπορεί να γράψει τις splines σαν γραμμικό συνδυασμό των πολυωνύμων Bernstein ορισμένες στο διάστημα κάθε στοιχείου. Έπειτα με βάση το θεωρητικό υπόβαθρο που θα έχει αναπτυχθεί θα δούμε πώς μπορούμε να προγραμματίσουμε στο Matlab έξυπνους αλγορίθμους για επίλυση ορισμένων υπολογιστικών προβλημάτων. Το πρώτο θα είναι η κατασκευή ενός αλγορίθμου T-splines ο οποίος θα μπορεί να εξάγει πίνακες διανυσμάτων και συναρτήσεις βάσης για οποιοδήποτε ορθογώνιο πλέγμα επιθυμούμε. Στη συνέχεια, το παράνω θα χρησιμοποιηθεί σε μία υπολογιστική εφαρμογή μετάδοσης θερμότητας σε ένα ορθογώνιο χωρίο. Τέλος, θα δώσουμε συμπεράσματα και αφορμές για συζήτηση με βάση τα αποτελέσματα που προέκυψαν από τους υπολογισμούς μας.

Abstract

This thesis aims to introduce the basic concepts of isogeometric analysis both theoretically and computationally. The theory to which we refer will contain sections such as B-splines, NURBS and T-splines which are based on B-splines and the concept of Bézier decomposition. Bézier decomposition can write the splines into Bernstein polynomials defined over the domain of one element. Then, based on the theoretical background that has been developed, we will see how we can program intelligent algorithms in Matlab to solve specific computational problems. The first one will be a construction of a T-splines algorithm that will be able to give an array of vectors and basis functions for any orthogonal grid we desire. Then, the above will be used to solve a computational problem of heat transfer in a rectangle. Finally, we will draw conclusions and discuss the results that have been derived from our previous calculations.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή Χριστόφορο Προβατίδη για την άψογη συνεργασία που είχαμε μέχρι την ολοκλήρωση αυτής της διπλωματικής αλλά και τον υποψήφιο διδάκτορα Ιωάννη Δημητρίου για την πολύτιμη συμβολή του. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου και τον αδερφό μου για την στήριξη τους όλο αυτό το διάστημα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Σκοπός αυτής της μεταπτυχιακής εργασίας είναι η κατασκευή στο προγραμματιστικό περιβάλλον Matlab ενός αλγορίθμου T-splines στις δύο διαστάσεις, που να ξεκινάει από ένα δοσμένο μη-δομημένο σύνολο κομβικών σημείων και να δίνει στην έξοδο τις συναρτήσεις μορφής και η μέτρηση της απόδοσης της εξαγωγής Bézier. Αυτή η εισαγωγή παρουσιάζει μερικές πληροφορίες και ιστορικά στοιχεία που θα αποτελέσουν το κίνητρο για την ολοκλήρωση της εργασίας.

1.1 CAE και CAD

Computer Aided Engineering (CAE) είναι η ευρεία χρήση λογισμικού ηλεκτρονικών υπολογιστών σε μηχανικά προβλήματα. Η μέθοδος που μας ενδιαφέρει κυρίως είναι η ανάλυση πεπερασμένων στοιχείων (FEA) ή οποία επινοήθηκε κυρίως από τον John Argyris (1955) [17] και τον Ray W. Clough (1960) [9]. Αυτή η μέθοδος διακριτικοποιεί το πρόβλημα σε μικρότερα στοιχεία, όπου εκεί λύνονται οι κύριες εξισώσεις. Έκτοτε, η υπολογιστική ισχύς έχει αυξηθεί σε τεράστιο βαθμό, και η ανάλυση πεπερασμένων στοιχείων μπορεί να εκτελεστεί παντού και αποτελεί υλικό εκμάθησης για μηχανικούς και επιστήμονες διαφόρων κλάδων.

Εκτός από το CAE εξίσου σημαντικό για επιστήμονες, μηχανικούς αλλά και σχεδιαστές είναι το Computer Aided Design (CAD). Τα συστήματα CAD που χρησιμοποιούνται σήμερα έχουν τις ρίζες τους από τη δουλειά δύο γάλλων επιστημόνων Pierre Bézier της Renault και Paul de Faget de Casteljau της Citroen [6] οι οποίοι δούλεψαν στις αντίστοιχες αυτοκινητοβιομηχανίες.

Το CAD είναι η χρήση υπολογιστών (ή workstations) για τη δημιουργία, τροποποίηση, ανάλυση ή βελτιστοποίηση ενός σχεδίου [12]. Το λογισμικό CAD χρησιμοποιείται για να αυξήσει την παραγωγικότητα του σχεδιαστή, να βελτιώσει την ποιότητα του σχεδιασμού, να βελτιώσει τις επικοινωνίες μέσω τεκμηρίωσης και να δημιουργήσει μια βάση δεδομένων για την κατασκευή [12]. Η έξοδος CAD είναι συχνά με τη μορφή ηλεκτρονικών αρχείων για εκτύπωση. Χρησιμοποιείται επίσης ο όρος CADD (Computer Aided Design and Drafting) [13], καθώς και ο Computer Aided Geometric Design (CAGD).

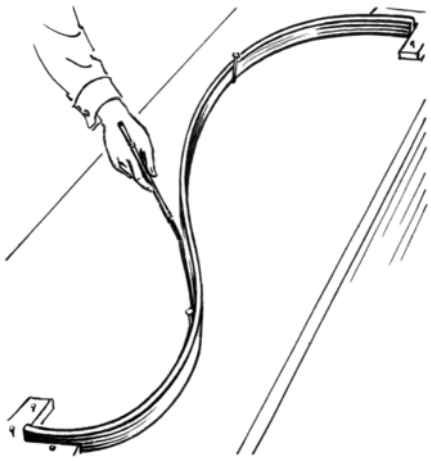
Το CAD χρησιμοποιείται ευρέως σε πολλούς τομείς της σύγχρονης βιομηχανίας, όπως οι αυτοκινητοβιομηχανίες, η ναυπηγική και η αεροδιαστημική. Το CAD χρησιμοποιείται επίσης ευρέως για την παραγωγή ειδικών εφέ σε ταινίες, διαφημιστικά κλπ. Επίσης μπορεί να χρησιμοποιηθεί για τον σχεδιασμό καμπυλών *(splines)* και σχημάτων σε δισδιάστατο χώρο όπου και θα το χρησιμοποιήσουμε ή επιφάνειες και

στερεά σε τρισδιάστατα γραφικά.

1.2 Splines

Στα γραφικά υπολογιστών και στο CAD, ο όρος spline συχνότερα αναφέρεται σε μια τμηματικά πολυωνυμική (παραμετρική) καμπύλη. Οι splines είναι δημοφιλείς καμπύλες σε αυτούς τους τομείς λόγω της απλότητας της κατασκευής τους, της ευκολίας και της ακρίβειας της αξιολόγησής τους και της ικανότητάς τους να προσεγγίζουν πολύπλοκα σχήματα μέσω της καμπύλης προσαρμογής και του σχεδιασμού της διαδραστικής καμπύλης.

Οι splines προέρχονται από τις εύκαμπτες κατασκευασμένες splines που χρησιμοποιούνται από τους ναυπηγούς και τους σχεδιαστές για να σχεδιάζουν ομαλά σχήματα (βλ. Σχήμα 1.1).



Σχήμα 1.1: Μία spline

Στα μαθηματικά, μια spline είναι μια ειδική συνάρτηση που ορίζεται τμηματικά από πολυώνυμα. Σε προβλήματα παρεμβολής, η παρεμβολή spline συχνά προτιμάται από την πολυωνυμική παρεμβολή, επειδή δίνει παρόμοια αποτελέσματα, ακόμα και όταν χρησιμοποιούνται πολυώνυμα χαμηλού βαθμού, αποφεύγοντας το φαινόμενο του Runge για υψηλότερους βαθμούς.

1.3 Φαινόμενο Runge

Στο μαθηματικό πεδίο της αριθμητικής ανάλυσης, το *φαινόμενο Runge (Runge's phenomenon)* (βλ. Σχήμα 1.2) είναι ένα πρόβλημα ταλάντωσης στις άκρες ενός διαστήματος που συμβαίνει όταν χρησιμοποιείται πολυωνυμική παρεμβολή με πολυώνυμα υψηλού βαθμού πάνω σε ένα σύνολο ισοδύναμων σημείων παρεμβολής. Η ανακάλυψη από τον Carl David Tolmé Runge (1901) [15] ήταν σημαντική επειδή δείχνει ότι η μετάβαση σε υψηλότερους βαθμούς δεν βελτιώνει πάντα την ακρίβεια.

Έστω λοιπόν η συνάρτηση Runge η οποία ορίζεται σαν

$$f(x) = \frac{1}{1 + 25x^2} \tag{1.1}$$

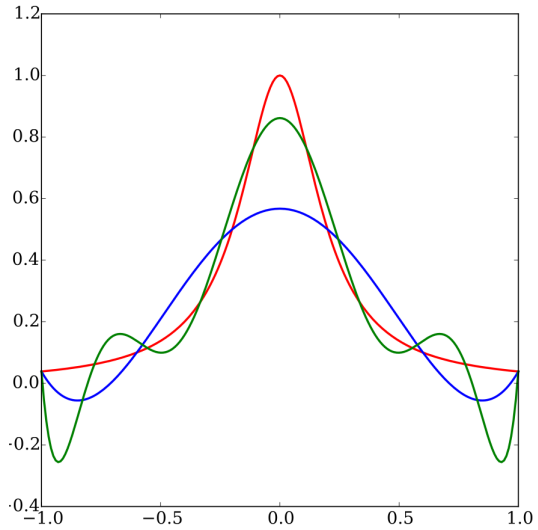
ο Runge διαπίστωσε ότι αν αυτή η συνάρτηση παρεμβάλλεται σε ίσα διαστήματα x_i μεταξύ -1 και 1 έτσι ώστε

$$x_i = \frac{2i}{n} - 1, \quad i \in \{0, 1, \ldots, n\} \tag{1.2}$$

με ένα πολυώνυμο $P_n(x)$ βαθμού $\geq n$, η προκύπτουσα παρεμβολή ταλαντώνεται προς το τέλος του διαστήματος, δηλαδή κοντά στο -1 και 1. Επίσης το σφάλμα της παρεμβολής αυξάνεται στο άπειρο όταν ο βαθμός του πολυωνύμου αυξάνεται δηλαδή

$$\lim_{n\to\infty} \left(\max_{-1\leq x\leq 1} |f(x) - P_n(x)| \right) = +\infty \tag{1.3}$$

Αυτό δείχνει ότι η πολυωνυμική παρεμβολή υψηλού βαθμού σε ισαπέχοντα σημεία μπορεί να φέρει μη επιθυμητά αποτελέσματα. Ένας τρόπος αυτό το φαινόμενο να αντιμετωπιστεί είναι χρησιμοποιώντας τμηματικά πολυωνυμικές καμπύλες spline.



Σχήμα 1.2: Το φαινόμενο Runge. Με κόκκινο απεικονίζεται η συνάρτηση Runge ενώ με πράσινο και μπλε πολυώνυμα παρεμβολής 5-ου και 9-ου βαθμού αντίστοιχα

1.4 Ισογεωμετρική Ανάλυση

Επειδή τα CAE και CAD αναπτύχθηκαν ανεξάρτητα, δεν είναι συμβατά. Η γεωμετρία περιγράφεται διαφορετικά, καθώς ο σκοπός και των δύο δεν ήταν ο ίδιος. Η ισογεωμετρική ανάλυση είναι ένας τρόπος για να ενσωματώσει αυτά τα δύο συστήματα και να επιτρέψει τα μοντέλα που σχεδιάζονται συνήθως στο CAD να χρησιμοποιηθούν στην FEA.

Η ισογεωμετρική ανάλυση εισήχθη από τον Tom Hughes [6, 11] στα πλαίσια της παρεμβολής NURBS. Ωστόσο, προηγούμενες προσπάθειες από το 1973 με χρήση παλαιότερων παρεμβολών CAGD (Coons, Gordon, Καρτεσιανών γινομένων πολυωνύμων Lagrange, κλπ.) έχουν καταγραφεί σε πρόσφατο βιβλίο [18].

Η ιδέα της ισογεωμετρικής ανάλυσης είναι να χρησιμοποιηθεί η ίδια βάση για την ανάλυση που χρησιμοποιείται στην περιγραφή της γεωμετρίας σε αντίθεση με την καθιερομένη μέθοδο των πεπερασμένων στοιχείων. Επιπλέον, τα ισογεωμετρικά στοιχεία

είναι επίσης ισοπαραμετρικά στοιχεία αφού η βάση της γεωμετρίας και της ανάλυσης είναι η ίδια. Η ισογεωμετρική ανάλυση χρησιμοποιεί Non-uniform rational basis splines (NURBS) (τη βάση των περισσότερων συστημάτων CAD) ή T-splines.

Η ισογεωμετρική ανάλυση έχει μελετηθεί εκτενώς και εφαρμόζεται σε πολλά διαφορετικά πεδία ανάλυσης όπως δομική ανάλυση, αλληλεπίδραση ρευστού-δομής και πεπερασμένη παραμόρφωση ελαστοπλαστικών στερεών.

1.5 Αυτή η μεταπτυχιακή εργασία

Αυτή η μεταπτυχιακή εργασία θα δώσει τις βασικές έννοιες για την κατανόηση της ισογεωμετρικής ανάλυσης τόσο σε θεωρητικό αλλά και σε υπολογιστικό επίπεδο.

Θα ξεκινήσει εισάγοντας στο Κεφάλαιο 2 τις B-splines και στο Κεφάλαιο 3 τις NURBS και έπειτα θα συνεχίσουμε επεκτείνοντας πάνω στις T-splines στο Κεφάλαιο 4 που αποτελούν μία σχετικά καινούργια τεχνική στο CAD [4, 11]. Στη συνέχεια αυτό το θεωρητικό υπόβαθρο θα μας βοηθήσει να κατανοήσουμε την εξαγωγή Bézier η οποία θα εξηγηθεί στο Κεφάλαιο 5. Τέλος στο Κεφάλαιο 6 θα αποτυπώσουμε την βασική μεθοδολογία της ισογεωμετρικής ανάλυσης και την σχέση της με την συνηθισμένη ανάλυση πεπερασμένων στοιχείων.

Έπειτα, όλο αυτό το θεωρητικό υπόβαθρο θα χρησιμοποιηθεί στο τέλος του ίδιου κεφαλαίου για να αναπτύξουμε στο προγραμματιστικό περιβάλλον Matlab έναν αλγόριθμο T-splines στις δύο διαστάσεις, που να ξεκινάει από ένα δοσμένο μη-δομημένο σύνολο κομβικών σημείων (knots) και να δίνει στην έξοδο τις συναρτήσεις μορφής. Στη συνέχεια ο κώδικας του προηγουμένου θα χρησιμοποιηθεί για τιν επίλυση ενός υπολογιστικού προβλήματος μετάδοσης θερμότητας σε ένα ορθογώνιο χωρίο. Τέλος, στο κεφάλαιο 7 θα γίνει μία ανασκόπηση των σημαντικών ευρημάτων και αποτελεσμάτων που προέκυψαν με σκοπό να καταλήξουμε σε ουσιώδη συμπεράσματα αλλά και αφορμές για περαιτέρω συζητήσεις και προβληματισμούς στο μέλλον.

Κατά την διάρκεια αυτής της εργασίας θα ¨περιοριστούμε¨ σε προβλήματα δύο χωρικών διαστάσεων. Η θεωρία που θα παρουσιαστεί εδώ θα είναι αρκετή για παραπάνω ενασχόληση στις τρεις διαστάσεις καθώς οι έννοιες είναι πιο ευκολονόητες στον διδιάστατο χώρο. Επίσης, θα προσπαθήσουμε με παραδείγματα, σχήματα, εικόνες αλλα και αριθμιτικούς υπολογισμούς να περιγράψουμε με όσο το δυνατόν περισσότερη σαφήνεια τα θέματα που θα παρουσιαστούν.

Κεφάλαιο 2

B-splines

Σ' αυτό το κεφάλαιο θα ορίσουμε τις B-splines, τις καμπύλες και τις επιφάνειες που σχετίζονται μ' αυτές καθώς και τις παραγώγους τους. Οι B-splines αποτελούν την βάση στην οποία θα στηριχτούμε τόσο για τις NURBS όσο και για τις T-splines οι οποίες θα συζητηθούν αργότερα.

2.1 Κομβοδιάνυσμα

Ένα σύνολο αυξανόμενων πραγματικών αριθμών που αντιπροσωπεύει συντεταγμένες στον χώρο που ορίζουν οι συναρτήσεις βάσης ονομάζεται *κομβοδιάνυσμα (knot vector)* και ορίζεται ως

$$U = \{u_1, ...u_{n+p+1}\} \tag{2.1}$$

με $u_i \leq u_{i+1}$, $i = 1, ..., n + p$ να είναι οι *κόμβοι (knots)*, p είναι ο βαθμός πολυωνύμου, n είναι ο αριθμός των συναρτήσεων βάσης και $|u| = n + p + 1$ το μήκος του.

Αξίζει να σημειωθεί πως οι κόμβοι μπορεί να είναι ή να μην είναι ισοδυνάμως κατανεμημένοι χωρικά που σημαίνει ότι πολλές τιμές κόμβων θα επαναλαμβάνονται. Η *πολλαπλότητα (multiplicity)* είναι ο βαθμός των επαναλαμβανόμενων κόμβων. Ένα κομβοδιάνυσμα είναι *ανοιχτό (open)* αν η πρώτη και η τελευταία τιμή εμφανίζονται $p + 1$ φορές. Ένα ανοιχτό κομβοδιάνυσμα σχηματίζει συναρτήσεις βάσης που παρεμβάλονται στα άκρα του χώρου και γι' αυτό χρησιμοιποιούνται στην κατασκευή συναρτήσεων βάσης στην ισογεωμετρική ανάλυση.

2.2 B-Splines

Υπάρχουν διάφοροι τρόποι να οριστούν οι συναρτήσεις B-splines και να αποδειχθούν οι σημαντικές ιδιότητές τους, π.χ. με διαιρεμένες διαφορές των αποκομμένων δυνάμεων, της μεθόδου συνένωσης σημείων και της αναδρομικής σχέσης λόγω των deBoor, Cox και Mansfield.

Χρησιμοποιούμε τον αναδρομικό τύπο δεδομένου ότι είναι ο πιο χρήσιμος και αποδοτικός για εφαρμογή σε ηλεκτρονικούς υπολογιστές. Η i-στή συνάρτηση βάσης B-spline βαθμού p (τάξης $p + 1$), που συμβολίζεται με $N_{i,p}(u)$, ορίζεται ως

$$N_{i,0}(u) = \begin{cases} 1 & \text{αν } u_i \leq u < u_{i+1} \\ 0 & \text{αλλού} \end{cases} \tag{2.2}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.3}$$

Αξίζει να σημειωθεί ότι [7]:

- Η συνάρτηση $N_{i,0}(u)$ είναι μια βηματική συνάρτηση, ίση με το μηδέν παντού εκτός από το ημιανοιχτό διάστημα $u \in [u_i, u_{i+1})$
- Για $p > 0$, τότε η $N_{i,p}(u)$ είναι γραμμικός συνδυασμός δύο συναρτήσεων βάσης βαθμού $(p-1)$
- Ο υπολογισμός ενός συνόλου βάσεων απαιτεί καθορισμό του κομβοδιανύσματος U και του βαθμού p
- Οι συναρτήσεις $N_{i,p}(u)$ είναι τμηματικά πολυώνυμα, που ορίζονται σε ολόκληρο τον άξονα των πραγματικών τιμών. Γενικά, έχουν ενδιαφέρον μόνο στο υποδιάστημα $[u_0, u_i]$
- Το ημι-ανοιχτό διάστημα $[u_i, u_{i+1})$ ονομάζεται i-στο *κομβοδιάστημα (knot span)*. Μπορεί να έχει μηδενικό μήκος, γιατί δεν απαιτείται οι κόμβοι να είναι διακριτοί
- Ο υπολογισμός των p-στων βαθμών συναρτήσεων δημιουργεί ένα τριγωνικό διάγραμμα
- Αν ο παρανομαστής μηδενίζεται τότε συμβαίνει το ίδιο και με το πηλίκο

2.2.1 Παράδειγμα συναρτήσεων βάσης

Για να κατανοήσουμε καλύτερα τις B-splines θα υπολογίσουμε και θα σχεδιάσουυμε τις συναρτήσεις βάσης για το κομβοδιάνυσμα $U = \{0, 0, 1, 1, 2, 3, 4, 4, 5\}$. Πρώτα θα υπολογίσουμε τις συναρτήσεις βάσης για $p = 0$ (σταθερές) στη συνέχεια για $p = 1$ (γραμμικές) και τέλος για $p = 2$ (τετραγωνικές). Έτσι ξεκινώντας για $p = 0$ έχουμε

$$N_{0,0}(u) = 0 \tag{2.4}$$

$$N_{1,0}(u) = \begin{cases} 1 & \text{αν } 0 \leq u < 1 \\ 0 & \text{αλλού} \end{cases} \tag{2.5}$$

$$N_{2,0}(u) = 0 \tag{2.6}$$

$$N_{3,0}(u) = \begin{cases} 1 & \text{αν } 1 \leq u < 2 \\ 0 & \text{αλλού} \end{cases} \tag{2.7}$$

$$N_{4,0}(u) = \begin{cases} 1 & \text{αν } 2 \leq u < 3 \\ 0 & \text{αλλού} \end{cases} \tag{2.8}$$

$$N_{5,0}(u) = \begin{cases} 1 & \text{αν } 3 \leq u < 4 \\ 0 & \text{αλλού} \end{cases} \tag{2.9}$$

$$N_{6,0}(u) = 0 \tag{2.10}$$

$$N_{7,0}(u) = \begin{cases} 1 & \text{αν } 4 \leq u < 5 \\ 0 & \text{αλλού} \end{cases} \tag{2.11}$$

τώρα εφόσον βρήκαμε τις συναρτήσεις βάσης για $p = 0$ μπορούμε να υπολογίσουμε τις $p = 1$

$$N_{0,1}(u) = \begin{cases} 1-u & \text{αν } 0 \le u < 1 \\ 0 & \text{αλλού} \end{cases} \tag{2.12}$$

$$N_{1,1}(u) = \begin{cases} u & \text{αν } 0 \le u < 1 \\ 0 & \text{αλλού} \end{cases} \tag{2.13}$$

$$N_{2,1}(u) = \begin{cases} 2-u & \text{αν } 1 \le u < 2 \\ 0 & \text{αλλού} \end{cases} \tag{2.14}$$

$$N_{3,1}(u) = \begin{cases} u-1 & \text{αν } 1 \le u < 2 \\ 3-u & \text{αν } 2 \le u < 3 \end{cases} \tag{2.15}$$

$$N_{4,1}(u) = \begin{cases} u-2 & \text{αν } 2 \le u < 3 \\ 4-u & \text{αν } 3 \le u < 4 \end{cases} \tag{2.16}$$

$$N_{5,1}(u) = \begin{cases} u-3 & \text{αν } 3 \le u < 4 \\ 0 & \text{αλλού} \end{cases} \tag{2.17}$$

$$N_{6,1}(u) = \begin{cases} 5-u & \text{αν } 4 \le u < 5 \\ 0 & \text{αλλού} \end{cases} \tag{2.18}$$

τέλος μπορούμε να υπολογίσουμε και τις συναρτήσεις βάσης για $p = 2$ εφόσον έχουμε τις $p = 1$

$$N_{0,2}(u) = \begin{cases} -2(u-1)u & \text{αν } 0 \le u < 1 \\ 0 & \text{αλλού} \end{cases} \tag{2.19}$$

$$N_{1,2}(u) = \begin{cases} u^2 & \text{αν } 0 \le u < 1 \\ (u-2)^2 & \text{αν } 1 \le u < 2 \end{cases} \tag{2.20}$$

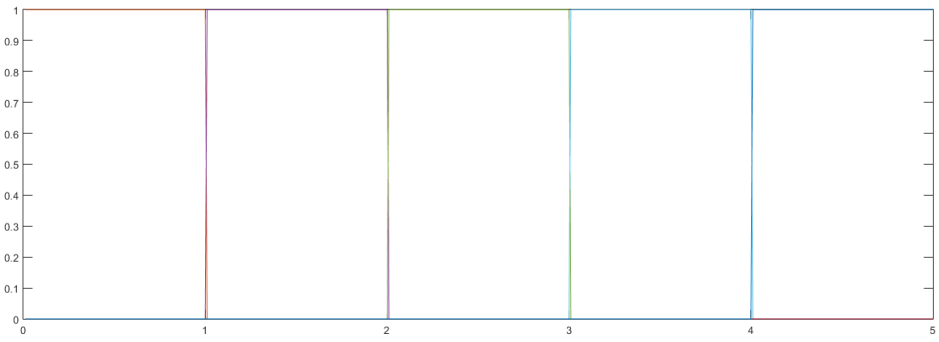
$$N_{2,2}(u) = \begin{cases} -\frac{3u^2}{2} + 5u - \frac{7}{2} & \text{αν } 1 \le u < 2 \\ \frac{1}{2}(u-3)^2 & \text{αν } 2 \le u < 3 \end{cases} \tag{2.21}$$

$$N_{3,2}(u) = \begin{cases} \frac{1}{2}(u-1)^2 & \text{αν } 1 \le u < 2 \\ -u^2 + 5u - \frac{11}{2} & \text{αν } 2 \le u < 3 \\ \frac{1}{2}(u-4)^2 & \text{αν } 3 \le u < 4 \end{cases} \tag{2.22}$$

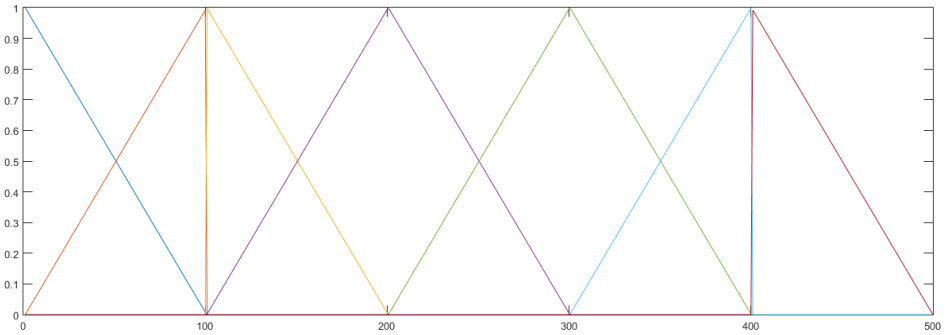
$$N_{4,2}(u) = \begin{cases} \frac{1}{2}(u-2)^2 & \text{αν } 2 \le u < 3 \\ -\frac{3u^2}{2} + 10u - 16 & \text{αν } 3 \le u < 4 \end{cases} \tag{2.23}$$

$$N_{5,2}(u) = \begin{cases} (u-3)^2 & \text{αν } 3 \le u < 4 \\ (u-5)^2 & \text{αν } 4 \le u < 5 \end{cases} \tag{2.24}$$

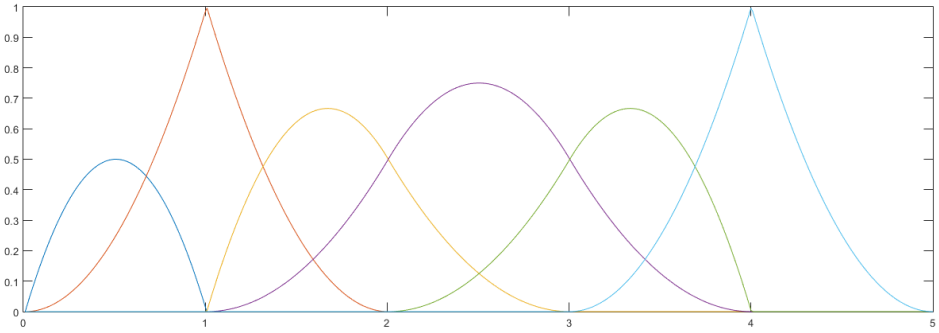
Στο Σχήμα 2.1 παρακάτω βλέπουμε τα γραφήματα των συναρτήσεων βάσης που μόλις υπολογίστηκαν.



(α΄) $p = 0$



(β΄) $p = 1$



(γ΄) $p = 2$

Σχήμα 2.1: Συναρτήσεις βάσης B-spline τάξης 0 έως 2 για $U = \{0, 0, 1, 1, 2, 3, 4, 4, 5\}$

2.3 Παράγωγοι συναρτήσεων B-splines

Με τη βοήθεια της μαθηματικής επαγωγής αποδεικνύεται ότι οι παράγωγοι των συναρτήσεων βάσης των B-splines δίνονται από την παρακάτω αναδρομική σχέση

$$N'_{i,p} = \frac{p}{u_{i+p} - u_i} N_{i,p-i}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.25}$$

Γενικότερα, εάν $N_{i,p}^{(k)}$ συμβολίζει την παράγωγο k-τάξης της $N_{i,p}$, συνεχόμενες παραγωγίσεις της (2.25) μας δίνουν τη γενική σχέση

$$N_{i,p}^{(k)}(u) = p\left(\frac{N_{i,p-1}^{(k-1)}}{u_{i+p} - u_i} - \frac{N_{i+1,p-1}^{(k-1)}}{u_{i+p+1} - u_{i+1}}\right) \tag{2.26}$$

Ένας εναλλακτικός τρόπος υπολογισμού της παραγώγου k-τάξης των συανρτήσεων $N_{i,p-k}, \ldots, N_{i+k,p-k}$ είναι ο ακόλουθος:

$$N_{i,p}^{(k)} = \frac{p!}{(p-k)!}\sum_{j=0}^{k} a_{k,j} N_{i+j,p-k} \tag{2.27}$$

όπου μερικές παρατηρήσεις για την παραπάνω εξίσωση

- Το k δεν θα πρέπει να είναι μεγαλύτερο του p
- Οι παρανομαστές που αφορούν διαφορές κόμβων μπορούν είναι μηδέν, το ποσοστό ορίζεται να είναι μηδέν σ' αυτήν την περίπτωση

$$\begin{aligned}
a_{0,0} &= 1 \\
a_{k,0} &= \frac{a_{k-1,0}}{u_{i+p-k+1} - u_i} \\
a_{k,j} &= \frac{a_{k-1,j} - a_{k-1,j-1}}{u_{i+p+j-k+1} - u_{i+j}} \quad j = 1, \ldots k-1 \\
a_{k,k} &= \frac{-a_{k-1,k-1}}{u_{i+p+1} - u_{i+k}}
\end{aligned}$$

2.4 Καμπύλες B-splines

Μία τμηματικά-πολυωνυμική καμπύλη B-spline p-στου βαθμού με $\mathbf{P}_i \in \mathbb{R}^d$ τα *σημεία ελέγχου (control points)* και $N_{i,p}$ οι γνωστές μας συναρτήσεις βάσεις για $i = 1, 2, \ldots, n$ που αναφέραμε πιο πάνω ορίζεται ως

$$C(u) = \sum_{i=1}^{n} N_{i,p}(u)\mathbf{P}_i \tag{2.28}$$

Αυτές οι καμπύλες B-spline όπως βλέπουμε κατασκευάζονται παρομοίως όπως και στην κλασική ανάλυση πεπερασμένων στοιχείων από ένα γραμμικό συνδυασμό συναρτήσεων βάσης αλλά στην προκειμένη περίπτωση χρησιμοποιούμε B-splines.

Το αξιοσημείωτο σχετικά με τις καμπύλες B-spline σε σχέση με τις καμπύλες στην καθιερομένη ανάλυση πεπερασμένων στοιχείων είναι ότι τα σημεία ελέγχου δεν είναι και αναγκαίως σημεία παρεμβολής όπως συμβαίνει στην ανάλυση πεπερασμένων στοιχείων. Δηλαδή παρόλο που τα σημεία ελέγχου είναι ανάλογα των κόμβων η καμπύλη συνήθως δεν παιρνάει από τα σημεία ελέγχου.

2.5 Επιφάνεις B-splines

Μία επιφάνεια B-spline κατασκευάζεται από συναρτήσεις βασης σε δύο κατευθύνσεις, $N_{i,p}(u)$ και $M_{j,q}(v)$ και ένα σύνολο σημείων ελέγχου $\mathbf{P}_{i,j}$ όπου $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$

με αντίστοιχα κομβοδιανύσματα $U = \{u_1, \ldots u_{n+p+1}\}$, $V = \{v_1, \ldots v_{m+q+1}\}$ και ορίζεται ως

$$S(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(u) M_{j,q}(v) \mathbf{P}_{i,j} \tag{2.29}$$

Δηλαδή μία επιφάνεια B-spline είναι αποτέλεσμα ενός γινομένου τανυστών.

Κεφάλαιο 3

NURBS

Οι NURBS (Non-Uniform Rational B-splines) μπορούν να κατανοηθούν με γεωμετρική και αλγεβρική διερεύνυση. Σ' αυτό το κεφάλαιο θα προσπαθήσουμε να παρουσιάσουμε την αλγεβρική δομή των καμπυλών και επιφανειών NURBS καθώς και να δώσουμε την βασική γεωμετρική τους ιδέα. Τέλος θα αναφερθούμε στην εισαγωγή κόμβων και στους δεσμούς που θα μας χρειαστούν στη συνέχεια.

3.1 Καμπύλες NURBS

Υπάρχουν γεωμετρικές οντότητες στον $\mathbb{R}^d$ που δεν μπορούν να μοντελοποιηθούν με ακρίβεια με τμηματικά πολυώνυμα. Πολλές σημαντικές, ωστόσο, μπορούν να κατασκευαστούν μέσω ενός προβολικού μετασχηματισμού μια αντίστοιχης B-spline στον $\mathbb{R}^{d+1}$.

Έτσι λοιπόν για να κατασκευάσουμε τις NURBS θα χρειαστούμε σημεία ελέγχου $\mathbf{P_i}$ που δίνονται από τον τύπο

$$(\mathbf{P}_i)_j = \frac{(\mathbf{P}_i^w)_j}{w_i} \qquad j = 1, \ldots, d \tag{3.1}$$

$$w_i = (\mathbf{P}_i^w)_{d+1} \tag{3.2}$$

όπου τα $w_i > 0$ είναι τα *βάρη (weights)* των σημείων ελέγχου και τα $\mathbf{P}_i^w$ τα προβολικά σημεία ελέγχου. Η συνάρτηση βάσης για μία καμπύλη NURBS ορίζεται ως

$$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=1}^{k} N_{j,p}(u)w_j} \tag{3.3}$$

με $W(u) = \sum_{i=1}^{n} N_{j,p}(u)w_j$ να είναι η *συνάρτηση βάρους (weighting function)* . Όμως η παραπάνω σχέση μπορεί να γραφτεί και σε μορφή πινάκων πολύ εύκολα

$$\mathbf{R}(u) = \frac{1}{W(u)}\mathbf{W}N(u) \tag{3.4}$$

όπου $\mathbf{W}$ διαγώνιος τετραγωνικός πίνακας και $N(u)$ το διάνυσμα στήλη συναρτήσεων βάσης της B-spline. Εφόσον έχουμε τις συναρτήσεις βάσης για τις NURBS μπορούμε εύκολα να κατασκευάσουμε την αντίστοιχη καμπύλη NURBS

$$C(u) = \sum_{i=1}^{n} R_{i,p}(u)\mathbf{P}_i \tag{3.5}$$

3.2 Επιφάνειες NURBS

Προφανώς εργαζόμενοι αναλόγως όπως και πρίν η συνάρτηση βάσης για μία επιφάνεια NURBS ορίζεται ως

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}} \tag{3.6}$$

με $W(u,v) = \sum_{l=1}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}$ η συνάρτηση βάρους. Και πάλι η παραπάνω σχέση μπορεί να γραφτεί σε μορφή πινάκων όπως και πριν ως

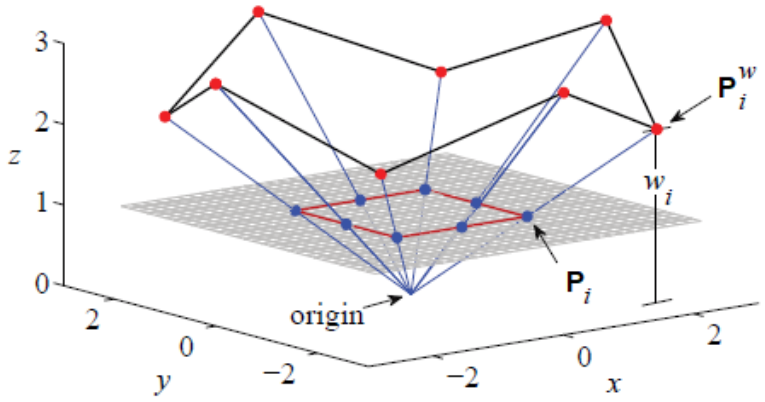
$$\mathbf{R}(u,v) = \frac{1}{W(u,v)}\mathbf{W}N(u)N(v) \tag{3.7}$$

όπου $\mathbf{W}$ πάλι ο διαγώνιος πίνακας βαρών και $N(u)$ και $N(v)$ τα διάνυσματα στήλες των B-spline συναρτήσεων βάσης. Εφόσον έχουμε τις συναρτήσεις βάσεις για τις NURBS μπορούμε εύκολα να κατασκευάσουμε την αντίστοιχη επιφάνεια NURBS

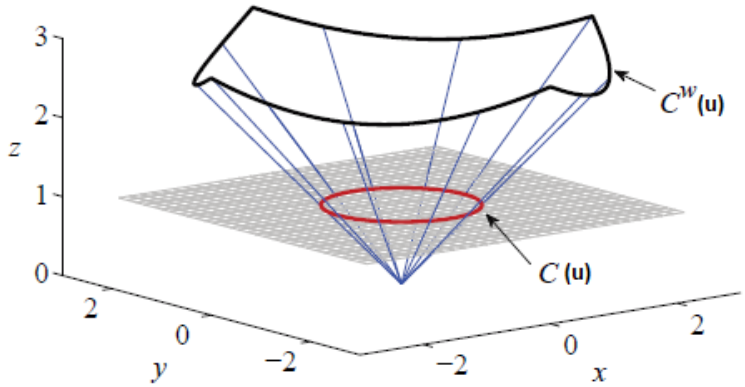
$$S(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}(u,v)\mathbf{P}_{i,j} \tag{3.8}$$

3.3 Γεωμετρική οπτική των NURBS

Από γεωμετρικής πλευράς μια NURBS στο $\mathbb{R}^d$ είναι το αποτέλεσμα ενός προβολικού μετασχηματισμού μιας B-Spline σε $\mathbb{R}^{d+1}$, όπου d είναι ο αριθμός των φυσικών διαστάσεων. Για παράδειγμα, μια τρισδιάστατη καμπύλη B-Spline που προβάλλεται στο επίπεδο $z = 1$ θα δημιουργήσει μια διδιάστατη καμπύλη NURBS (βλ. Σχήμα 3.1).



(α') Ο προβολικός μετασχηματισμός του προβολικού σημείου ελέγχου $\mathbf{P}_i^w$ μας δίνει το $\mathbf{P}_i$



(β') Ο προβολικός μετασχηματισμός της B-Spline καμπύλης $C^w(u)$ μας δίνει την καμπύλη NURBS $C(u)$

Σχήμα 3.1: Ένας κύκλος κατασκευασμένος στον $\mathbb{R}^2$ από τον προβολικό μετασχηματισμό μιας B-Spline στον $\mathbb{R}^3$ [6]

3.4 Εισαγωγή κόμβων

Καινούργιοι κόμβοι μπορεί να εισαχθούν στα γνωστά μας κομβοδιανύσματα U, V χωρίς να αλλάξουμε την καμπύλη B-Spline. Αυτή η συγκεκριμένη διαδικασία που μας δίνει καινούργια σημεία ελέγχου $\bar{\mathbf{P}}$ αποκαλείται *εισαγωγή κόμβων (knot insertion)* [3]. Οι εξισώσεις που διέπουν αυτή την διαδικασία είναι οι

$$\bar{P}_A = \begin{cases} P_A & A = 1 \\ a_A P_A + (1 - a_A) P_{A-1} & 1 < A < m \\ P_n & A = m \end{cases} \tag{3.9}$$

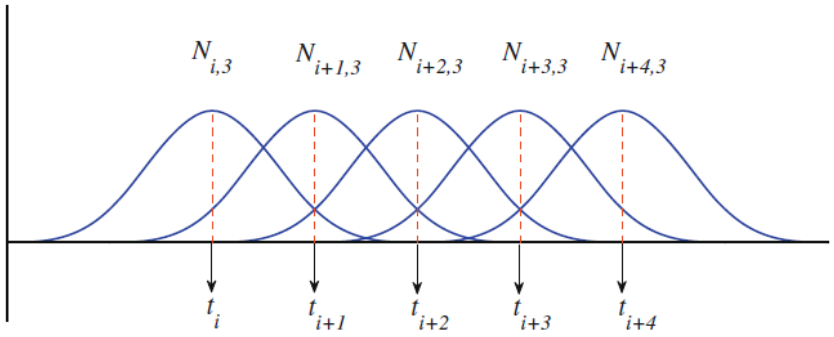
$$a_A = \begin{cases} 1 & 1 \le A \le k - p \\ \frac{\bar{u} - u_A}{u_{A+p} - u_A} & k - p + 1 \le A \le k \\ 0 & A \ge k + 1 \end{cases} \tag{3.10}$$

3.5 Δεσμοί

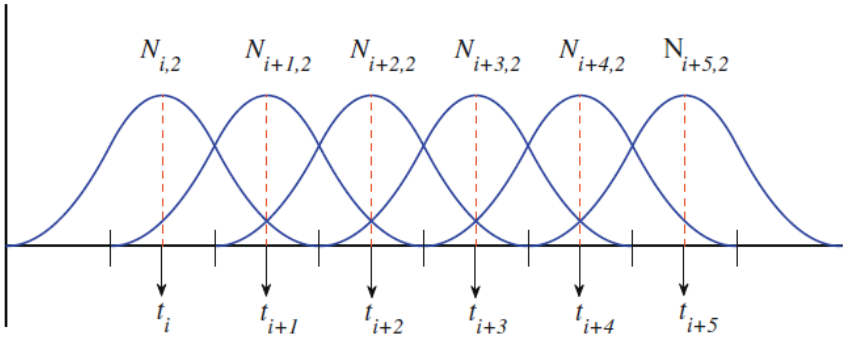
Στην ισογεωμετρική ανάλυση, ο αριθμός των συναρτήσεων βάσης δεν εξαρτάται από των αριθμό των κόμβων. Γι΄ αυτό το λόγο λοιπόν είναι βολικό να εισάγουμε τους *δεσμούς (anchors)* t_i για κάθε συνάρτηση βάσης για να έχουμε μία τοποθεσία στον χώρο που να σχετίζεται με την συγκεκριμένη βάση. Η εξίσωση των δεσμών ορίζεται ως

$$t_i = \begin{cases} u_{i+\frac{p+1}{2}} & \text{αν p περιττό} \\ \frac{1}{2}(u_{i+\frac{p+1}{2}} + u_{i+\frac{p+1}{2}+1}) & \text{αν p άρτιο} \end{cases} \tag{3.11}$$

Αυτό σημαίνει πως για άρτιο βαθμού πολυώνυμο ο δεσμός θα βρίσκεται στη μέση του κομβοδιαστήματος ενώ για περιττού βαθμού θα βρίσκεται πάνω στους κόμβους (βλ. Σχήμα 3.2). Προφανώς οι B-Splines δεν είναι απαραίτητο να οριστούν μέσων των δεσμών αλλά θα φανούν χρήσιμοι στη συνέχεια.



(α΄) Περιττός βαθμός, οι δεσμοί είναι πάνω στους κόμβους



(β΄) Άρτιος βαθμός, οι δεσμοί είναι στο μέσο των κομβοδιαστημάτων

Σχήμα 3.2: Δεσμοί για τις περιπτώσεις περιττού και άρτιου βαθμού [16]

Κεφάλαιο 4

T-splines

Οι T-splines εισήχθησαν αρχικά από τον Sederberg το 2003 [11] κυρίως ως μια γενίκευση των NURBS που αναφέραμε στο προήγουμενο κεφάλαιο. Το κύριο πλεονέκτημα των T-splines είναι ότι δεν είναι συνδεδεμένες με διαφόρους περιορισμούς στην δομή τους όπως συμβάινει με το το τανυστικό γινόμενο των NURBS. Σ' αυτό το κεφάλαιο θα προσπαθήσουμε να εισάγουμε την έννοια των T-splines καθώς και τις ιδιότητες τους αφού πρώτα ορίσουμε τις PB-splines που διώχνουν τους περιορισμούς του τανυστικού γινομένου.

4.1 PB-Splines

Όπως είδαμε και πριν μια επιφάνεια B-Spline προκύπτει από το χώρο που ορίζουν τα αντιστοίχα κομβοδιανύσματα U και V. Οι PB-Splines είναι επιφάνειες που τα σημεία ελέγχου τους δεν έχουν καμία τοπολογική σχέση μεταξύ τους και έτσι αποτελούνται από διάφορες συναρτήσεις, τις λεγόμενες ως *blending functions*. Οι PB-Splines δεν σχετίζονται με ένα πλέγμα αλλά με ανεξάρτητα σημεία και *τοπικά κομβοδιανύσματα (local knot vectors)*, πράγμα που δεν συμβαίνει με τις B-Splines συναρτήσεις βάσης οι οποίες εξαρτώνται αποκλειστικά από ένα ολικό κομβοδιάνυσμα.

Έτσι λοιπόν παρακάμπτοντας τον περιορισμό ενός ενιαίου κομβοδιανύσματος και απλά μία PB-Spline να αποτελείται από ένα σύνολο τοπικών κομβοδιανυσμάτων U_i, V_i τα οποία με τη σειρά τους θα αντιστοιχουν στις συναρτήσεις βάσης B-Spline N_i, θα έχουμε την εξίσωση μίας PB-Spline σαν

$$B(u,v) = \frac{\sum_{i=1}^{n} \mathbf{P_i} N_i(u,v)}{\sum_{i=1}^{n} N_i(u,v)} \qquad (u,v) \in \mathbf{D} \tag{4.1}$$

όπου $\mathbf{D}$ ο χώρος που ορίζεται η PB-Spline, $\mathbf{P_i}$ τα γνωστά μας σημεία ελέγχου και $\sum_{i=1}^{n} N_i(u,v) > 0$.

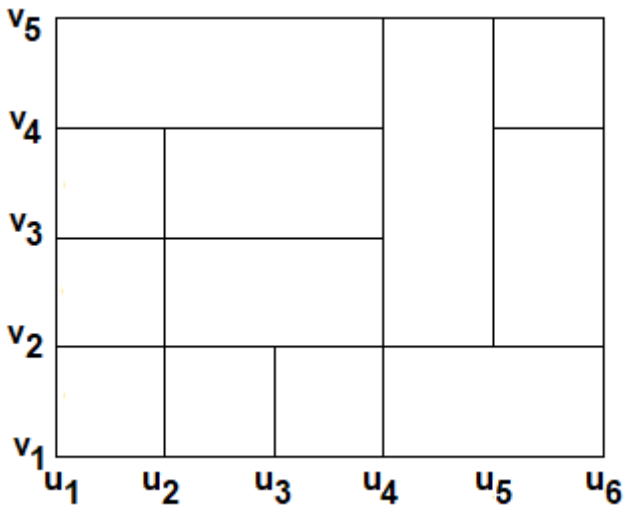
Αξίζει να σημειώσουμε πως τα τοπικά κομβοδιανύσματα U_i, V_i αν και είναι ανεξάρτητα μεταξύ τους μπορεί να περιλαμβάνουν τις ίδιες τιμές των κόμβων.

4.2 T-splines

Οι συναρτήσεις T-splines είναι στην ουσία PB-splines με κάποιους περιορισμούς στα σημεία ελέγχου. Οι T-splines μπορούν να δημιουργήσουν έναν μή ομοιογενές πλέγμα που αναφέρεται ως *T-mesh* και αντιπροσωπεύει είτε τον *παραμετρικό χώρο (parametric*

space) (u, v) στην περίπτωση μας (βλ. Σχήμα 4.1) είτε το πλέγμα των σημείων ελέγχου που είναι ο *φυσικός χώρος (physical space)*.

Το T-mesh στον παραμετρικό χώρο (u, v) ουσιαστικά είναι μια εικονική αναπαράσταση του φυσικού χώρου. Είναι ένα ορθογώνιο πλέγμα που δεν είναι αναγκαίο να έχει ολοκληρωμένες γραμμές και στήλες δημιουργώντας έτσι τις λεγόμενες *T-junctions*. Τα σύμβολα $\vdash$, $\dashv$, $\perp$ και $\top$ είναι οι τέσσερεις πιθανοί σχηματισμοι των T-junctions που προκύπτουν σ'ένα συμβολικό T-mesh (βλ. Σχήμα 4.1).

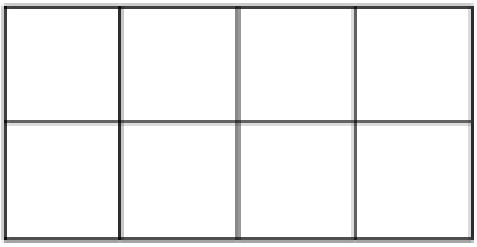


Σχήμα 4.1: Εικόνα ενός T-mesh

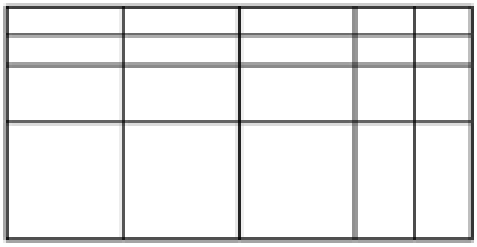
Τα πλεονεκτήματα όταν χρησιμοποιούμε T-splines είναι πολλά. Κατ' αρχάς, τα τοπικά κομβοδιανύσματα U_i, V_i προκύπτουν για κάθε βάση από το T-mesh και είναι απολύτως συμβατές με τις NURBS. Τα πλεονεκτήματα των T-splines σε αντίθεση με τις NURBS όμως δεν σταματούν εδώ. Τα σήμαντικότερα από αυτά ακολοθούν παρακάτω σε μορφή σχηματικών απεικονίσεων για καλύτερη κατανόηση.

- Μας επιτρέπουν να *τροποποιήσουμε-βελτιώσουμε (local refinement)* το πλέγμα μας όχι απλά ολικά, αλλά και τοπικά (βλ. Σχήμα 4.2).

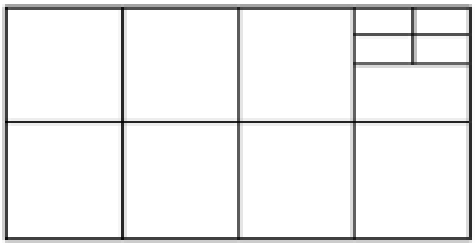
  Στο Σχήμα 4.2α βλέπουμε ένα αρχικό ορθογώνιο πλέγμα, στη συνέχεια στο Σχήμα 4.2β βλέπουμε πως υποχρεούμαστε με τις NURBS να έχουμε ολοκληρωμένες γραμμές και στήλες όπως είπαμε και πιο πάνω και τέλος στο Σχήμα 4.2γ βλέπουμε πως μπορούμε με την βοηθεία των T-splines να έχουμε πολύ μεγαλυτερη ελευθερία βελτίωσης στο πλέγμα δημιουργώντας T-junctions.



(α΄) Χωρίς βελτίωση



(β΄) Ολική NURBS βελτίωση



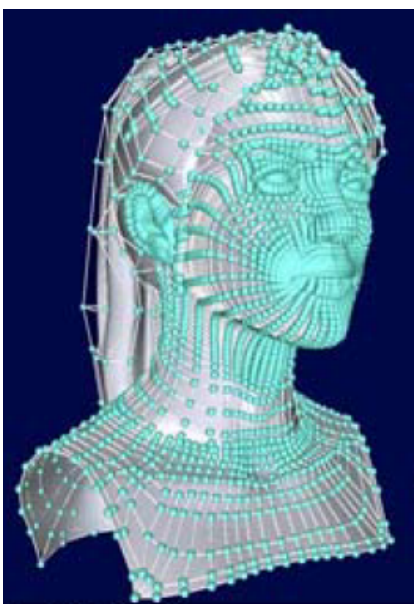
(γ΄) Τοπική T-spline βελτίωση

Σχήμα 4.2: Βελτίωση πλέγματος

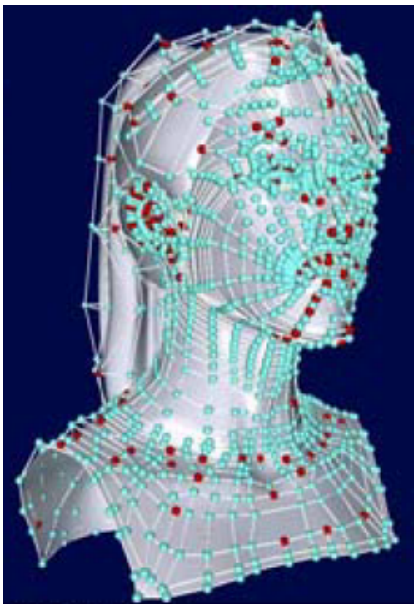
- Μας επιτρέπουν να χρησιμοποιήσουμε λιγότερα σημεία ελέγχου.

  Παρακάτω βλέπουμε ένα μοντέλο γυναίκας στο Σχήμα 4.3α που αρχικά μοντελοποιήθηκε με NURBS στο Σχήμα 4.3β και στη συνέχεια με T-splines στο Σχήμα 4.3γ μειώνοντας έτσι δραστικά τον αριθμό των σημείων ελέγχου κατά 61.62%

(α') Αρχικό μοντέλο γυναίκας



(β') NURBS 10305 σημεία ελέγχου



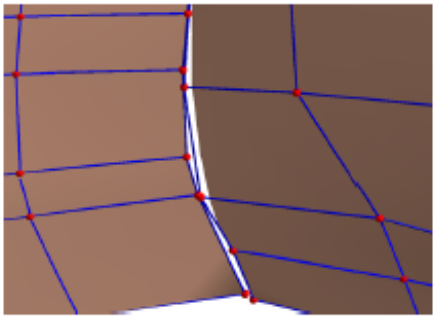
(γ') T-spline 3955 σημεία ελέγχου

Σχήμα 4.3: Μοντέλο γυναίκας [4]

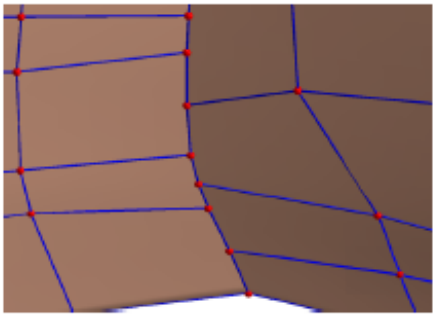
- Τέλος μας δίνουν την δυνατότητα να ¨κολλάμε¨ επιφάνειες με σχετική ευκολία εάν προκύψουν τυχόν κενά μεταξύ τους.

  Στο Σχήμα 4.4α βλέπουμε ένα μοντέλο χεριού που αποτελείται από επιφάνειες NURBS. Έπειτα βλέπουμε στο Σχήμα 4.4β αφού το μεγενθύνουμε το κένο που προκύπτει μεταξύ των δύο επιφανειών εφόσον χρησιμοποιήσαμε NURBS . Τέλος βλέπουμε στο Σχήμα 4.4γ πως αυτό το πρόβλημα διορθώνεται με την βοήθεια των T-splines και το κενό παύει να υπάρχει.

(α΄) Χέρι μοντελοποιημένο με NURBS



(β΄) Μεγένθυση NURBS



(γ΄) Μεγένθυση T-splines

Σχήμα 4.4: Το κενό κλείνει με την χρήση T-spline [11]

4.3 Κομβοδιαστήματα και τοπικά κομβοδιανύσματα

Οι πληροφορίες σχετικά με τα κομβοδιανύσματα για τις T-splines εκφράζονται χρησιμοποιώντας κομβοδιαστήματα. Τα κομβοδιαστήματα είναι θετικοί αριθμοί d_i, e_i που μας υποδηλώνουν τη διαφορά μεταξύ των δύο κόμβων. Το κάθε κομβοδιάστημα απαραιτήτως διέπεται από τους παρακάτω δύο κανόνες [11]:

- Το άθροισμα των κομβοδιαστημάτων των αντιθέτων πλευρών της κάθε έδρας πρέπει να είναι πάντα ίσο.
- Αν μία T-junction σε μία πλευρά μίας έδρας μπορεί να συνδεθεί με μία άλλη T-junction σε μία αντίθετη πλευρά της έδρας χωρίς να παραβιάζει τον πρώτο κανόνα, τότε αυτή η πλευρά θα πρέπει να συμπεριληφθεί στην T-junction.

Όπως αναφέραμε και πριν μπορούμε να συμπεράνουμε ποια είναι τα τοπικά κομβοδιανύσματα U_i, V_i για κάθε συνάρτηση βάσης N_i από το T-mesh. Για να προκύψει ένα σύστημα συντεταγμένων από κόμβους πρώτα επιλέγουμε ένα αρχικό σημείο ελέγχου σαν την αρχή των παραμετρικών μας συντεταγμένων (u, v). Ας πούμε λοιπόν ότι η αρχή των αξόνων μας είναι το $(0, 0)$ και ότι κάθε κομβοδιάστημα έχει μήκος d

$$d = u_i - u_{i-1} = v_j - v_{j-1} \tag{4.2}$$

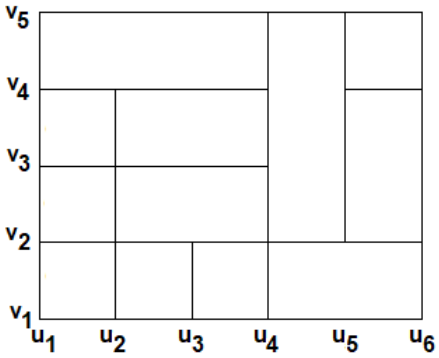
με $i = 1, \ldots, 6$ και $j = 1, \ldots, 5$.

Βέβαια στην περίπτωση μας που το το πλέγμα μας είναι ομοιογενές, δηλαδή τα κομβοδιαστήματα είναι ίσα μεταξύ τους, η δουλειά μας γίνεται κάπως ευκολότερη. Έτσι μπορούμε να γράψουμε τα σημεία ελέγχου ως $\mathbf{P_1} = (u_4, v_4)$ και $\mathbf{P_2} = (u_5 + d/2, v_3)$.

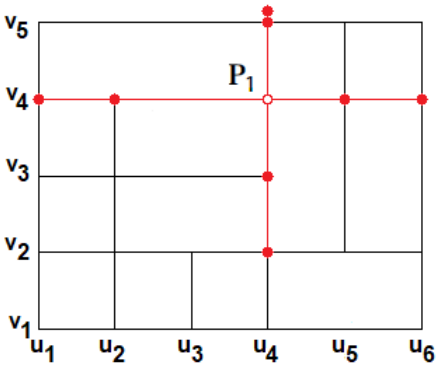
Τώρα για να προκύψουν τα τοπικά κομβοδιανύσματα των $\mathbf{P_1}$, $\mathbf{P_2}$ μπορούμε να χρησιμοποιήσουμε μία πολύ απλή μέθοδο εισάγοντας μία *παράμετρο ακτίνας (ray parameter)* $r(a) = (u_i + a, v_i)$ [11] όπου στην ουσία κρατάμε τις συντεταγμένες u_i του σημείου ελέγχου $\mathbf{P_i}$ και στη συνέχεια βαδίζουμε αριστερά και δέξιά από αυτό επιλέγοντας τα πιο κοντινά σ' αυτό σημεία. Αντιστοίχως ακολουθούμε την ίδια μεθοδολογία για να βρούμε τα v_i βαδίζοντας κατακόρυφα.

Έτσι λοιπόν, σύμφωνα με όσα είπαμε παραπάνω, θα έχουμε για το $\mathbf{P_1}$ τα $U_1 = \{u_1, u_2, u_4, u_5, u_6\}$ και $V_1 = \{v_2, u_3, u_4, v_5, v_5\}$ ενώ θα έχουμε για το $\mathbf{P_2}$ τα $U_2 = \{u_4, u_5, u_6, u_6\}$ και $V_2 = \{v_1, v_2, v_4, v_5\}$ για $p = 3$ και $p = 2$ αντίστοιχα. Το μήκος των τοπικών κομβοδιανυσμάτων θα πρέπει να έιναι ίσο με $l = p + 2$.

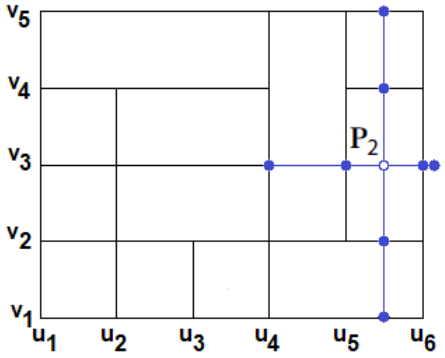
Αξίζει να σημειώσουμε δύο πολύ σημαντικές λεπτομέρειες. Πρώτον, αν οι συνοριακές γραμμές ξεπεραστούν η τελευταία τιμή επαναλαμβάνεται μέχρι να γεμίσουν οι θέσεις του κομβοδιανύσματος και δεύτερον εφόσον δέν βρίσκεται πάνω σε κάποια γραμμή η τιμή του παραλείπεται. Παρακάτω στο Σχήμα 4.5 μπορούμε να δούμε τον πλήρη συλλογισμό που μόλις αναπτύχθηκε.



(α') Αρχικό T-mesh



(β') Σημείο ελέγχου $\mathbf{P_1}$ για $p = 3$



(γ') Σημείο ελέγχου $\mathbf{P_2}$ για $p = 2$

Σχήμα 4.5: Εύρεση των τοπικών κομβοδιανυσμάτων από το T-mesh

Η παραπάνω διαδικασία εύρεσης των τοπικών κομβοδιανυσμάτων για τα $\mathbf{P_1}$, $\mathbf{P_2}$ μπορεί να γίνει χρησιμοποιώντας τα σημεία δεσμούς που είχαμε εισάγει στο προηγούμενο κεφάλαιο.

Ας πάρουμε την περίπτωση για την περίπτωση περιττού βαθμού πολυωνύμου $p = 3$ ο δεσμός είναι πάνω στον κόμβο. Σύμφωνα με τον τύπο για τους δεσμούς θα έχουμε ότι $t_i = u_{i+2}$. Τώρα για να βρούμε τα τοπικά κομβοδιανύσματα ακολουθώντας την

ίδια διαδικασία περπατώντας $(p+1)/2$ σε κάθε κατεύθυνση μέχρι να συναντήσουμε τα πλησιέστερα σημεία. Οπότε τα τοπικά κομβοδιανύσματα για το $t_1 = (u_4, v_4)$ θα είναι και πάλι τα $U_1 = \{u_1, u_2, u_4, u_5, u_6\}$ και $V_1 = \{v_2, u_3, u_4, v_5, v_5\}$.

Τώρα για την περίπτωση άρτιου βαθμού πολυωνύμου $p = 2$ ο δεσμός βρίσκεται στο κέντρο του κομβοδιαστήματος άρα θα έχουμε $t_i = \frac{1}{2}(u_{i+1} + u_{i+2})$. Βαδίζουμε και πάλι $p/2 + 1$ σε κάθε κατεύθυνση από το σημείο $t_2 = (u_5 + d/2, v_3)$ και έτσι τα τοπικά κομβοδιανύσματα θα είναι πάλι τα $U_2 = \{u_4, u_5, u_6, u_6\}$ και $V_2 = \{v_1, v_2, v_4, v_5\}$.

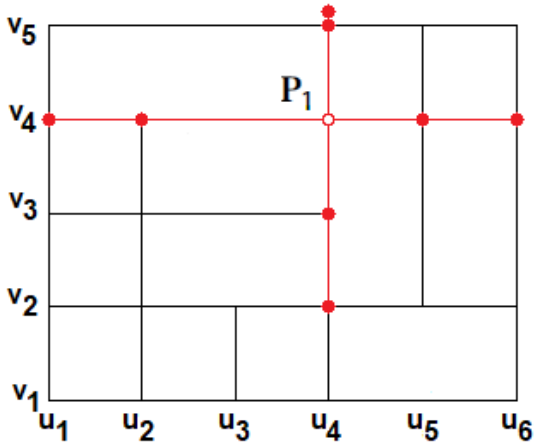
Εφόσον αυτά τα τοπικά κομβοδιανύσματα έχουν καταγραφεί για κάθε συνάρτηση βάσης, τότε η T-spline ορίζεται χρησιμοποιώντας την εξίσωση για τις PB-spline.

4.4 Επεκταμένο T-mesh

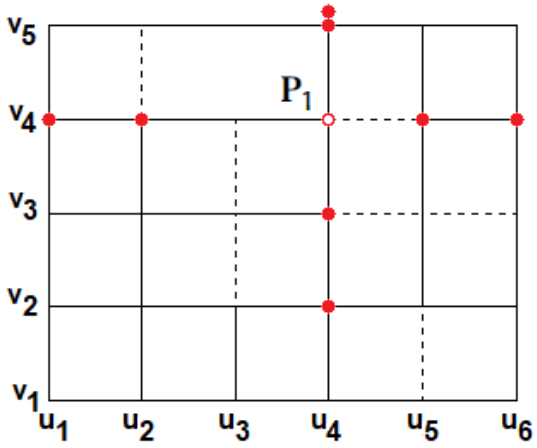
Ένα T-mesh περιέχει επιπλέον γραμμές μειωμένης συνέχειας σε αντίθεση με ένα NURBS-mesh που η μείωση της συνέχειας παρατηρείται μόνο στις γραμμές κόμβων. Αυτές οι γραμμές επεκτείνουν τις γραμμές κόμβων από τις T-junctions. Οι γραμμές αυτές συμβολίζονται συνήθως ως διακεκομμένες γραμμές. Αυτό το T-mesh που περιλαμβάνει τις γραμμές μειωμένης συνέχειας καλείται *επεκταμένο (extended)* T-mesh και πάνω σ' αυτό τα στοιχεία των T-splines ορίζονται. Τα στοιχεία των T-splines είναι ορθογώνιες περιοχές πάνω στις οποίες οι συναρτήσεις βάσης των T-splines είναι $\mathbf{C}^\infty$.

Οι επεκταμένες T-junctions σχηματίζονται βαδίζοντας προς τις T-junctions στην κατεύθυνση κατα την οποία λείπουν πλευρές, μέχρι να σχηματιστούν $(p+1)/2$ κάθετες πλευρές. Αυτή η εύκολη μέθοδος μπορεί να χρησιμοποιθεί για περιττούς πολυωνυμικούς βαθμούς. Για άρτια p οι γραμμές μειωμένης συνέχειας θα εμφανιστούν σ' όλες τις τιμές της αντίστοιχης τοπικής συνάρτησης βάσης της T-spline. Εάν δεν υπάρχει ήδη μία γραμμή κόμβου στο πλέγμα μας, τότε οι ήδη υπάρχουσες γραμμές κόμβων πρέπει να επεκταθούν στην περιοχή στήριξης της συνάρτησης βάσης. Επαναλαμβάνοντας αυτή την διαδικασία για όλους τους κόμβους δημιουργούμε το επεκταμένο T-mesh.

Στο Σχήμα 4.6 παρακάτω βλέπουμε το επεκταμένο T-mesh που σχηματίστηκε σύμφωνα με την παραπάνω μέθοδο. Στην αρχή βλέπουμε τα αρχικό T-mesh πάνω στον κόμβο $\mathbf{P_1}$ για $p = 3$ στο Σχήματα 4.6α και στη συνέχεια την επέκτασή του στο Σχήμα 4.6β Αναλόγως εργαζόμαστε και για άρτια p.



(α΄) Δεσμός για $p = 3$



(β΄) Επεκταμένο T-mesh για $p = 3$

Σχήμα 4.6: Γραμμές μειωμένης συνέχειας

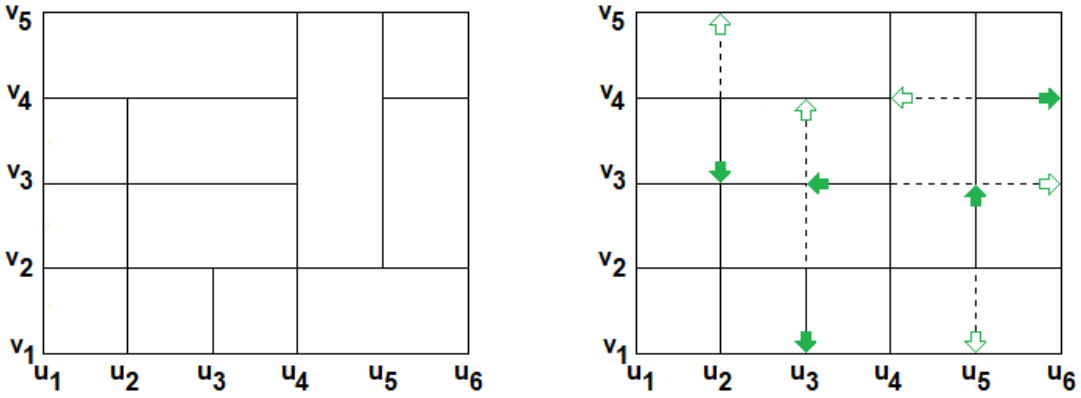
4.5 Κατάλληλες για ανάλυση T-splines

Κατάλληλες για ανάλυση T-splines (Analysis-suitable T-splines) είναι στην ουσία ένα υποσύνολο των T-splines. Πολλές σημαντικές ιδιότητες έχουν αναπτυχθεί για αυτές. Η πιο σημαντική από αυτές είναι ότι οι συναρτήσεις που απαρτίζουν τις T-splines (blending functions) είναι γραμμικά ανεξάρτητες για κάθε επιλογή κόμβων. Επιπλέον ιδιότητες που τις διέπουν είναι η κυρτότητα πολυγώνου (convex hull), θετικές συναρτήσεις βάσης, τοπική βελτίωση, γενίκευση σε όποιον βαθμό θέλουμε κ.α. [5]

Μία κατάλληλη για ανάλυση T-spline είναι μία T-spline που το T-mesh της είναι κατάλληλο για ανάλυση. Ένα T-mesh που είναι κατάλληλο για ανάλυση σύμφωνα με τον Scotts θα είναι όταν ¨οι επεκτάσεις των T-junctions που σχηματίζονται δεν θα διασταυρώνονται, ακόμη και το τελικό σημείο τομής θεωρείται ως παραβίαση του κανόνα T-junction¨ [14].

Στα Σχήματα 4.7 και 4.8 που ακολουθούν θα δούμε παραδείγματα T-meshes που το πρώτο παραβιάζει τον κανόνα της καταλληλότητας για ανάλυση και στη συνέχεια το δεύτερο που κατασκευάστηκε με σκοπό να είναι κατάλληλο για ανάλυση.

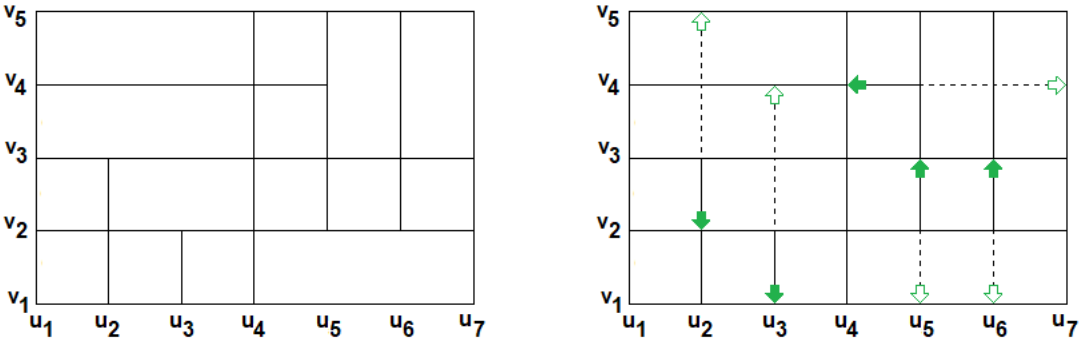
Στο παρακάτω Σχήμα 4.7α το T-mesh που χρησιμοποιήσαμε και πριν δεν είναι κατάλληλο για ανάλυση διότι οι προεκτάσεις των T-junction για $p = 3$ τέμνονται σε ορισμένα σημεία στο Σχήμα 4.7β



(α') T-mesh (β') Επεκταμένο T-mesh για $p = 3$

Σχήμα 4.7: Ακατάλληλο για ανάλυση T-mesh

Ένα κατάλληλο για ανάλυση T-mesh βρίσκεται παρακάτω στο Σχήμα 4.8α και το επεκταμένο μας για $p = 3$ T-mesh στο Σχήμα 4.8β.



(α') ASTS T-mesh (β') Επεκταμένο T-mesh για $p = 3$

Σχήμα 4.8: Κατάλληλο για ανάλυση (ASTS) T-mesh

Κεφάλαιο 5

Εξαγωγή Bézier

Η εξαγωγή Bézier είναι ένα σημαντικό εργαλείο που διευκολύνει την ενσωμάτωση των NURBS αλλά και των T-splines σε οποιοδήποτε κώδικα πεπερασμένων στοιχείων. Η ιδέα είναι βασισμένη στο ότι κάθε συνάρτηση B-spline μπορεί να γραφτεί ως γραμμικός συνδυασμός των πολυωνύμων Bernstein. Σ' αυτό το κεφάλαιο λοιπόν θα αναφερθούμε πρώτα στις καμπύλες Bézier και στα πολυώνυμα Bernstein και ύστερα θα προσπαθήσουμε να αποτυπώσουμε την θεωρητική βάση της εξαγωγής Bézier. Σημαντικό ποσοστό της θεωρίας αυτού του κεφαλαίου καλύφθηκέ από τις [1, 2].

5.1 Πολυώνυμα Bernstein και καμπύλες Bézier

Το σύνολο των πολυωνύμων $\mathbf{B}(u) = \{\mathbf{B}_{i,p}(u)\}_i^{p+1}$ που ορίζεται συνήθως στο διάστημα $[0, 1]$ και σχηματίζεται από ένα σύνολο σημείων ελέγχου με τιμές κόμβων $\mathbf{P} = \{\mathbf{P_i}\}_i^{p+1}$ με $\mathbf{P_i} \in \mathbb{R}^d$ ονομάζεται σύνολο πολυωνύμων Bernstein . Τότε τα πολυώνυμα Bernstein ορίζονται αναδρομικά σαν

$$B_{i,p}(u) = (1-u)B_{i,p-1}(u) + uB_{i-1,p-1}(u) \qquad u \in [0,1] \tag{5.1}$$

με

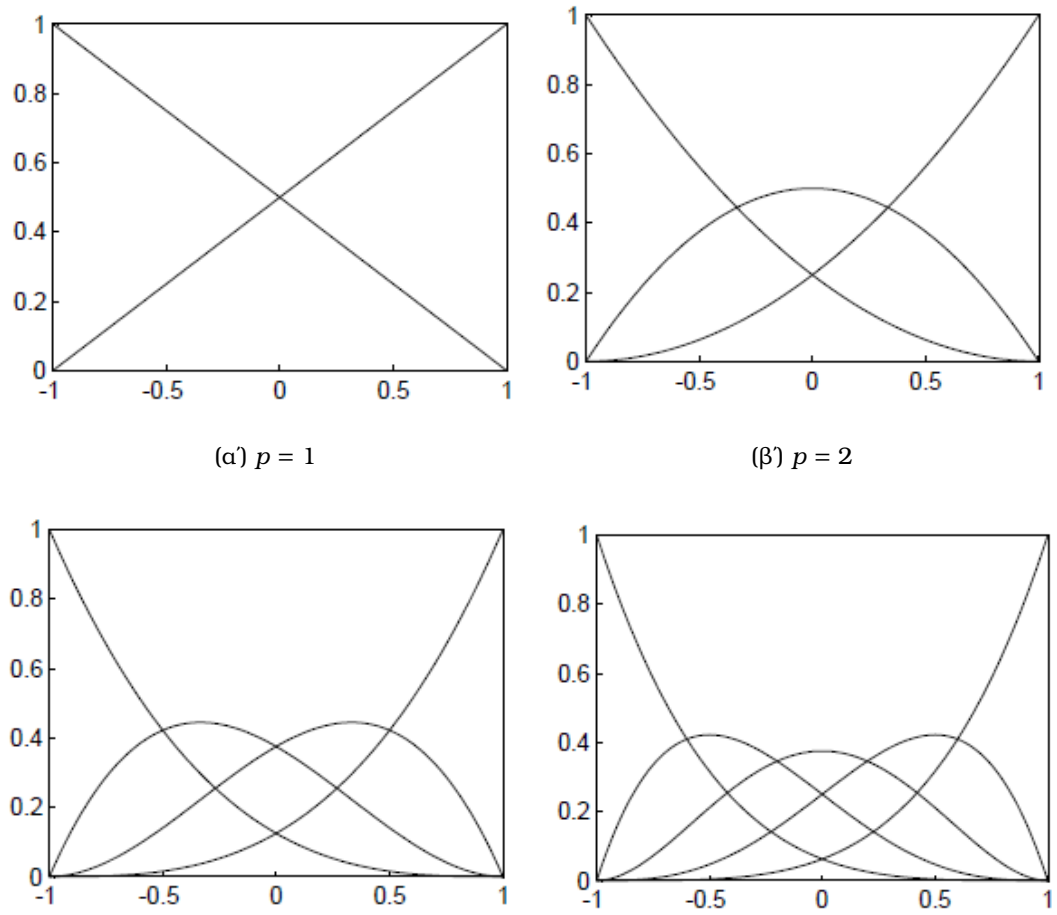
$$\begin{aligned} &B_{i,p}(u) = 0 \quad \text{αν } i < 1 \text{ ή } i > p+1 \\ &B_{i,p}(u) = 1 \end{aligned} \tag{5.2}$$

και έχουν πολλές σημαντικές ιδιότητες όπως:

- $B_{i,p}(u) \geq$ για $0 \leq u \leq 1$ (θετικότητα)
- $B_{i,p}(u) = B_{p-i,p}(1-u)$ (συμμετρία)
- $\sum_{i=1}^{p+1} B_{i,p}(u) = 1$ (κανονικοποίηση)
- $i^i p^{(-p)}(p-i)^{(p-i)}\binom{p}{i}$ (μοναδικό τοπικό μέγιστο)

Τα πολυώνυμα Bernstein μπορούν να σχηματιστούν χρησιμοποιώντας τις εξισώσεις των B-splines. Όμως, αφού οι τιμές των κόμβων είναι ορισμένες στο διάστημα $[-1, 1]$ με την κάθε μια να επαναλαμβάνεται $(p+1)$ φορές μπορούμε να γράψουμε τα πολυώνυμα Bernstein (βλ. Σχήμα 5.1) στο διάστημα $[-1, 1]$ σαν

$$B_{i,p}(u) = \frac{1}{2}(1-u)B_{i,p-1}(u) + \frac{1}{2}(1+u)B_{i-1,p-1}(u) \qquad u \in [-1,1] \tag{5.3}$$



(α') $p = 1$ (β') $p = 2$

(γ') $p = 3$ (δ') $p = 4$

Σχήμα 5.1: Πολυώνυμα Bernstein

και τις αντίστοιχες παραγώγους τους ως

$$\frac{d}{du}B_{i,p}(u) = \frac{p}{2}(B_{i-1,p-1}(u) - B_{i,p-1}(u)) \tag{5.4}$$

Τέλος, παραθέτουμε και την εξίσωση της καμπύλης Bézier η οποία είναι ένας γραμμικός συνδυασμός των πολυωνύμων Bernstein και ορίζεται ως

$$C(u) = \sum_{i=1}^{p+1} B_{i,p}(u)\mathbf{P}_i = \mathbf{P}_i^T\mathbf{B}(u) \tag{5.5}$$

με τον $\mathbf{P}$ να είναι ένας πίνακας $n \times d$, με d τον αριθμό των φυσικών διαστάσεων.

5.2 Αποσύζευξη Bézier

Ο τελεστής της εξαγωγής Bézier προβάλλει γραμμικούς συνδυασμούς των πολυωνύμων Bernstein σε μία βάση συναρτήσεων B-spline. Αυτός ο μετασχηματισμός μας δίνει τη δυνατότητα να χρησιμοποιήσουμε τμηματικά C^0 Bézier στοιχεία στην ισογεωμετρική ανάλυση. Για να εξάγουμε τα στοιχεία Bézier ακολουθούμε μία διαδικασία κατά την οποία ένα σύνολο B-splines συναρτήσεων βάσης μας δίνουν τα Bézier στοιχεία αυτών.

Αυτή η διαδικασία καλείται *αποσύζευξη (decomposition) Bézier* και επιτυγχάνεται με την εισαγωγή επιπλέον κόμβων στους εσωτερικούς μας κόμβους μέχρι η πολλαπλότητα του κάθε κόμβου να είναι ίση με p.

Έστω τώρα το γνωστό μας κομβοδιάνυσμα $U = \{u_0, \dots u_{n+p+1}\}$ με τα αντοίστοιχα σημεία ελέγχου $\{\mathbf{P}_A\}_A^n$ για την καμπύλη B-spline. Επίσης, έστω ένα άλλο σύνολο κόμβων $\bar{U} = \{\bar{u}_1, \dots \bar{u}_m\}$ όπου οι κόμβοι του $\bar{u}_j$ για $j = 0, \dots, m$ χρειάζονται για να μας δώσουν την αποσύζευξη Bézier της B-spline. Έτσι, για κάθε καινούργιο κόμβο $\bar{u}_j$ που θα εισάγεται παίρνουμε $a_A^j, A = 1, 2, \dots, n+j$ σύμφωνα με τις εξισώσεις της εισαγωγής κόμβου. Τώρα ορίζοντας τον $\mathbf{C}_j \in \mathbf{R}^{(n+j-1)\times(n+j)}$ σαν

$$\mathbf{C}^j = \begin{bmatrix} a_1 & 1-a_2 & 0 & \dots & & & 0 \\ 0 & a_2 & 1-a_3 & 0 & \dots & & 0 \\ 0 & 0 & a_3 & 1-a_4 & 0 & \dots & 0 \\ \vdots & & & & \ddots & & \\ 0 & \dots & & & 0 & a_{n+j-1} & 1-a_{n+j} \end{bmatrix} \tag{5.6}$$

Τα καινούργια σημεία ελέγχου υπολογίζονται αναδρομικά

$$\bar{\mathbf{P}}^{j+1} = (\mathbf{C}^j)^T \mathbf{P}^j \tag{5.7}$$

με $\bar{\mathbf{P}}^1 = \mathbf{P}$. Ορίζοντας ως

$$\mathbf{C}^T = (\mathbf{C}^m)^T (\mathbf{C}^{m-1})^T \dots (\mathbf{C}^1)^T \tag{5.8}$$

και $\bar{\mathbf{P}}^{m+1} = \mathbf{P}^b$ ως τα τελικά σημεία ελέγχου θα έχουμε τα

$$\mathbf{P}^b = \mathbf{C}^T \mathbf{P} \tag{5.9}$$

που είναι η σχέση που συνδέει τα κανούργια σημεία ελέγχου Bézier με τα κλασσικά σημεία ελέγχου των B-spline.

Αξίζει να σημειώσουμε πως το $\mathbf{P}$ έχει διαστάστεις $n \times 2$, το $\mathbf{C}$ διαστάσεις $n \times (n+m)$ και το $\mathbf{P}^b$ διαστάσεις $(n+m) \times 2$ σ' ένα d-διάστατο φυσικό χώρο.

Εφόσον η αποσύζευξη Bézier δεν αλλάζει την γεωμετρική ή παραμετρική ταυτότητα της καμπύλης τότε μπορεί να γραφτεί σαν

$$C(u) = (\mathbf{P}^b)^T \mathbf{B}(u) = (\mathbf{C}^T \mathbf{P})^T \mathbf{B}(u) = \mathbf{P}^T \mathbf{C}\mathbf{B}(u) = \mathbf{P}^T \mathbf{N}(u) \tag{5.10}$$

έτσι μας δίνεται η σχέση που συνδέει τα πολυώνυμα Bernstein με τις συναρτήσεις βάσης B-spline ως

$$\mathbf{N}(u) = \mathbf{C}\mathbf{B}(u) \tag{5.11}$$

με $\mathbf{C}$ να ονομάζεται ο τελεστής *εξαγωγής* Bézier *(Bézier extraction operator)*.

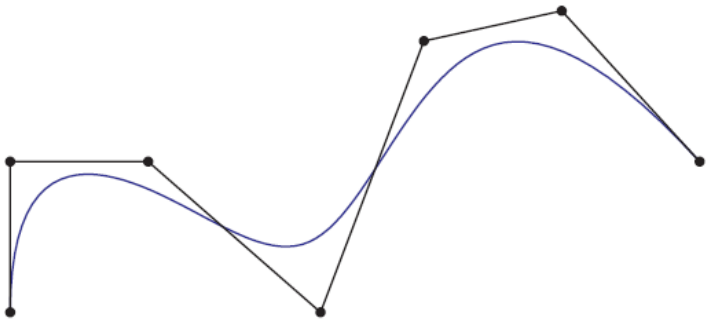
Αξίζει να σημειωθεί πως το μόνο που χρειάζεται για να κατασκευαστεί ο $\mathbf{C}$ είναι το κομβοδιάνυσμα, που σημαίνει πως ο τελεστής εξαγωγής Bézier δεν εξαρτάται από τα σημεία ελέγχου ή από συναρτήσεις βάσεις.

Έστω, για μία κυβική B-spline το αντίστοιχο κομβοδιάνυσμα της

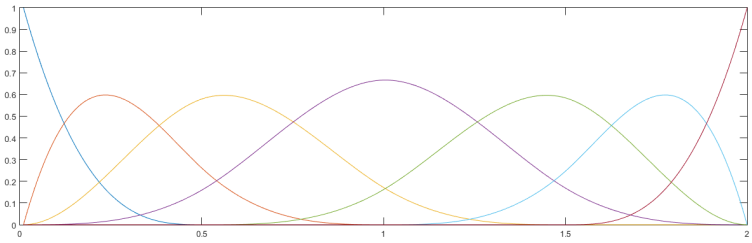
$$U = \{0, 0, 0, 0, 0.5, 1, 1.5, 2, 2, 2, 2\} \tag{5.12}$$

Για να φτιάξουμε τα Bézier στοιχεία της καμπύλης όπως είπαμε παραπάνω πραγματοποιούμε εισαγωγή κόμβων σε όλους τους εσωτερικούς μας κόμβους μέχρι η πολλαπλότητα τους να είναι ίση με $p = 3$, δηλαδή τον βαθμό της κυβικής καμπύλης. Έτσι, θα εισάγουμε τους κόμβους $\{0.5, 0.5, 1, 1, 1.5, 1.5\}$ έναν έναν στο U.

Στο Σχήμα 5.2 παρακάτω βλέπουμε την αρχική μας καμπύλη και τις συναρτήσεις βάσης της πριν την εισαγωγή κόμβων ενώ στο Σχήμα 5.3 παρατηρούμε την διαδικασία



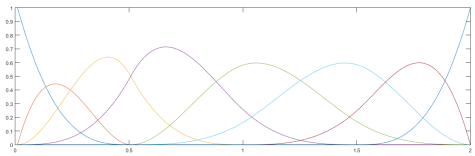
(α΄) Κυβική καμπύλη



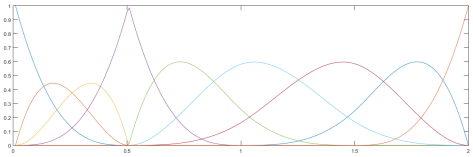
(β΄) Συναρτήσεις βάσης

Σχήμα 5.2: Η κυβική καμπύλη (α) και οι συναρτήσεις βάσης της (β)

εισαγωγής κόμβων βήμα προς βήμα και τέλος βλέπουμε στο Σχήμα 5.4 μετά την ολο-

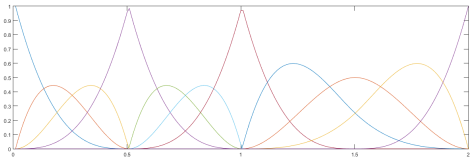


(α΄) Εισαγωγή 0.5

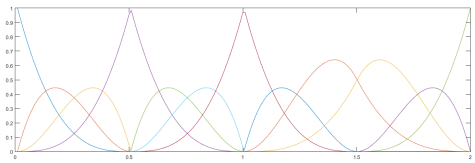


(β΄) Εισαγωγή 0.5

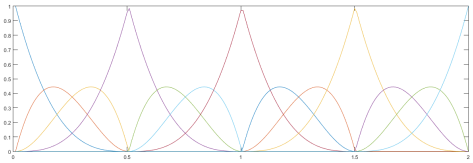
(γ΄) Εισαγωγή 1



(δ΄) Εισαγωγή 1



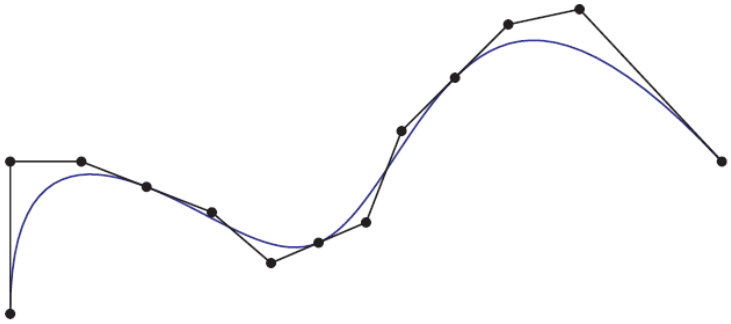
(ε΄) Εισαγωγή 1.5



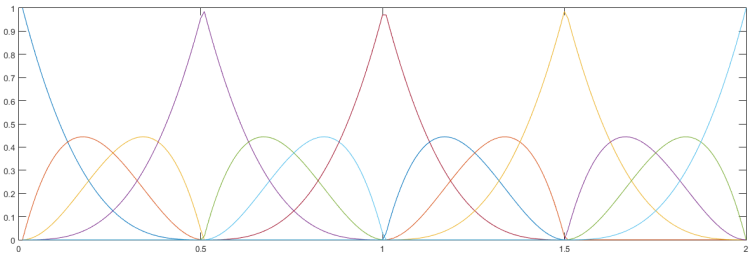
(ς΄) Εισαγωγή 1.5

Σχήμα 5.3: Η διαδικασία της εισαγωγής των κόμβων $\{0.5, 0.5, 1, 1, 1.5, 1.5\}$ στο κομβοδιάνυσμα U

κλήρωση της εισαγωγής όλων των κόμβων ότι η καμπύλη παρέμεινε ουσιαστικά η ίδια και το τελικό μας σύνολο συναρτήσεων βάσης Bézier.



(α') Κυβική καμπύλη



(β') Συναρτήσεις βάσης

Σχήμα 5.4: Η κυβική καμπύλη (α) και το τελικό σύνολο συναρτήσεων βάσης Bézier (β) μετά την εισαγωγή κόμβων

5.3 Εξαγωγή Bézier των NURBS

Όπως αναφέραμε και στο Κεφάλαιο 3, οι NURBS κατασκευάζονται χρησιμοποιώντας συναρτήσεις βάσης B-spline, έτσι μας επιτρέπεται να χρησιμοποιήσουμε τον τελεστή εξαγωγής στις NURBS. Αρχίζουμε ορίζωντας την συνάρτηση βάρους ως

$$\begin{aligned} W(u) &= \sum_{i=1}^{n} N_{i,p}(u) w_i = \mathbf{w}^T \mathbf{N}(u) \\ &= \mathbf{w}^T \mathbf{C} \mathbf{B}(u) = (\mathbf{C}^T \mathbf{w})^T \mathbf{B}(u) \\ &= (\mathbf{w}^b)^T \mathbf{B}(u) = W^b(u) \end{aligned} \tag{5.13}$$

όπου τα $\mathbf{w}^b = \mathbf{C}^T \mathbf{w}$ είναι τα βάρη Bézier. Χρησιμοποιώντας τις εξισώσεις του τελεστή εξαγωγής και τις συναρτήσεις βάσης για την καμπύλη NURBS θα έχουμε

$$\mathbf{R}(u) = \frac{1}{W^b(u)} \mathbf{W} \mathbf{C} \mathbf{B}(u) \tag{5.14}$$

όπου με $\mathbf{W}$ είναι τα βάρη NURBS.

Όπως και με την εισαγωγή κόμβων, η αποσύζευξη Bézier των σημείων ελέγχου γίνεται απευθείας στην καμπύλη B-spline που ορίζει την καμπύλη NURBS. Γεωμετρικώς αυτό επιτυγχάνεται προβάλλοντας τα σημεία ελέγχου της NURBS σε $d+1$ διαστάσεις, μετά ο τελεστής εξαγωγής Bézier εφαρμόζεται στα σημεία ελέγχου της B-spline και μετά η καμπύλη προβάλλεται πίσω στις d διαστάσεις για να πάρουμε τα στοιχεία Bézier, $\mathbf{P}^b$. Ορίζουμε τον διαγώνιο πίνακα $\mathbf{W}^b$ που αποτελείται από τα βάρη Bézier ως

$$\mathbf{W}^b = w_i \delta_{i,j} \tag{5.15}$$

Τώρα μπορούμε να υπολογίσουμε τα σημεία ελέγχου $\mathbf{P}^b$ της NURBS σαν

$$\mathbf{P}^b = (\mathbf{W}^b)^{-1}\mathbf{C}^T\mathbf{W}\mathbf{P} \tag{5.16}$$

και πολλαπλασιάζοντας με $\mathbf{W}^b$ θα έχουμε

$$\mathbf{W}^b\mathbf{P}^b = \mathbf{C}^T\mathbf{W}\mathbf{P} \tag{5.17}$$

Τέλος συνδυάζοντας τα παραπάνω μπορούμε να πάρουμε την καμπύλη NURBS με βάση τα στοιχεία Bézier σαν

$$C(u) = \frac{1}{W^b(u)}(\mathbf{W}^b\mathbf{P}^b)^T\mathbf{B}(u) = \sum_{i=1}^{n+n}\frac{\mathbf{P}_i^b w_i^b B_i(u)}{W^b(u)} \tag{5.18}$$

Τέλος για μία επιφάνεια ο διπλός τελεστής εξαγωγής που μας χρειάζεται ορίζεται για ένα στοιχείο ως

$$\mathbf{C}_A^e = \mathbf{C}_v^i \otimes \mathbf{C}_u^j \tag{5.19}$$

όπου με $\otimes$ είναι το γνωστό μας γινόμενο τελεστών και ορίζεται ως

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{1,1}\mathbf{B} & A_{1,1}\mathbf{B} & \dots \\ A_{2,1}\mathbf{B} & A_{2,2}\mathbf{B} & \\ \vdots & & \ddots \end{bmatrix} \tag{5.20}$$

5.3.1 Παράδειγμα της αποσύζευξης Bézier

Για το κομβοδιάνυσμα $U = \{0, 0, 0, 0, 0.5, 1, 1.5, 2, 2, 2, 2\}$ της καμπύλης Bézier που χρησιμοποιήσαμε πιο πριν μπορούμε να υπολογίσουμε τους τελεστές εξαγωγής $\mathbf{C}$, έτσι λοιπόν $\mathbf{N} = \mathbf{C}\mathbf{B}$ θα έχουμε

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} = \begin{bmatrix} C^1 & & \mathbf{0} & \\ & C^2 & & \\ & & C^3 & \\ \mathbf{0} & & & C^4 \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \\ B_8 \\ B_9 \\ B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix} \tag{5.21}$$

όπου $\mathbf{C_{7\times13}}$ ο γενικός πίνακανας των τελεστών εξαγωγής που αν και δεν είναι αναγκαίο να υπολογιστεί είναι χρήσιμος για την κατανόηση της διαδικασίας. Ο $\mathbf{C_{7\times13}}$ αποτελείται από τους επιμέρους τελεστές εξαγωγής $\mathbf{C}^1, \mathbf{C}^2, \mathbf{C}^3, \mathbf{C}^4$ που αποτελούν για το κάθε διάστημα $[0, 0.5), [0.5, 1), [1, 1, 5), [1.5, 2)$ αντίστοιχα τα στοιχεία Bézier. Έτσι, για το κάθε διάστημα του αρχικού μας κομβοδιαστήματος η αρχική βάση των NURBS μπορεί

να αναπαρασταθεί από έναν γραμμικό συνδυασμό συναρτήσεων βάσης του στοιχείου Bézier. Αναλυτικότερα θα έχουμε

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{Bmatrix} \tag{5.22}$$

$$\begin{Bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{Bmatrix} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 7/12 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{Bmatrix} \tag{5.23}$$

$$\begin{Bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \begin{Bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{Bmatrix} \tag{5.24}$$

$$\begin{Bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix} \tag{5.25}$$

Τώρα για να εντοπίσουμε τον εκάστοτε τελεστή εξαγωγής πρώτα εντοπίζουμε της συναρτήσεις βάσης για το κάθε στοιχείο έτσι ώστε

$$\begin{Bmatrix} N_1^1 \\ N_2^1 \\ N_3^1 \\ N_4^1 \end{Bmatrix} = \begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix}, \qquad \begin{Bmatrix} N_1^2 \\ N_2^2 \\ N_3^2 \\ N_4^2 \end{Bmatrix} = \begin{Bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{Bmatrix} \tag{5.26}$$

$$\begin{Bmatrix} N_1^3 \\ N_2^3 \\ N_3^3 \\ N_4^3 \end{Bmatrix} = \begin{Bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix}, \qquad \begin{Bmatrix} N_1^4 \\ N_2^4 \\ N_3^4 \\ N_4^4 \end{Bmatrix} = \begin{Bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} \tag{5.27}$$

και

$$\begin{Bmatrix} B_1^1 \\ B_2^1 \\ B_3^1 \\ B_4^1 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{Bmatrix}, \qquad \begin{Bmatrix} B_1^2 \\ B_2^2 \\ B_3^2 \\ B_4^2 \end{Bmatrix} = \begin{Bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{Bmatrix} \tag{5.28}$$

$$\begin{Bmatrix} B_1^3 \\ B_2^3 \\ B_3^3 \\ B_4^3 \end{Bmatrix} = \begin{Bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{Bmatrix}, \qquad \begin{Bmatrix} B_1^4 \\ B_2^4 \\ B_3^4 \\ B_4^4 \end{Bmatrix} = \begin{Bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix} \tag{5.29}$$

όπου οι πάνω δείκτες συμβολίζουν τον αριθμό του στοιχείου $e = 1, 2, 3, 4$. Τώρα με τις τοπικές ποσότητες πλέον να έχουν οριστεί μπορούμε να κατασκευάσουμε τους τελεστές εξαγωγής για κάθε στοιχείο, δηλαδή με μια απλή αντικατάσταση θα έχουμε

$$\begin{Bmatrix} N_1^1 \\ N_2^1 \\ N_3^1 \\ N_4^1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_1^1 \\ B_2^1 \\ B_3^1 \\ B_4^1 \end{Bmatrix} \tag{5.30}$$

$$\begin{Bmatrix} N_1^2 \\ N_2^2 \\ N_3^2 \\ N_4^2 \end{Bmatrix} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 7/12 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_1^2 \\ B_2^2 \\ B_3^2 \\ B_4^2 \end{Bmatrix} \tag{5.31}$$

$$\begin{Bmatrix} N_1^3 \\ N_2^3 \\ N_3^3 \\ N_4^3 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \begin{Bmatrix} B_1^3 \\ B_2^3 \\ B_3^3 \\ B_4^3 \end{Bmatrix} \tag{5.32}$$

$$\begin{Bmatrix} N_1^4 \\ N_2^4 \\ N_3^4 \\ N_4^4 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_1^4 \\ B_2^4 \\ B_3^4 \\ B_4^4 \end{Bmatrix} \tag{5.33}$$

Έτσι, η αρχική μας εξίσωση $\mathbf{N} = \mathbf{CB}$ μπορεί να γραφτεί για το κάθε στοιχείο ξεχωρισά ως

$$\mathbf{N}^e = \mathbf{C}^e\mathbf{B}^e \tag{5.34}$$

5.4 Εξαγωγή Bézier των T-splines

Στην περίπτωση των NURBS, ο τελεστής εξαγωγής υπολογίστηκε για ολόκληρα τα στοιχεία, όμως για τις T-splines η διαδικασία που ακολουθείται είναι λίγο διαφορετική αλλά η κεντρική ιδέα είναι ίδια. Έτσι λοιπόν, στην περίπτωση των T-splines, κάθε μία από αυτές θα εξάγεται ανεξάρτητα και θα προσθέτει μία γραμμή στον τελεστή εξαγωγής Bézier. Για τις T-splines τα τοπικά κομβοδιανύσματα είναι γενικώς ανοιχτά και τα *επεκταμένα κομβοδιανύσματα (extended knot vectors)* μπορούν να μας βοηθήσουν στην προκειμένη περίπτωση.

Γενικά, ο κανόνας που ισχύει για την κατασκευή των επεκταμένων κομβοδιανυσμάτων είναι ότι δημιουργούνται από τα τοπικά κομδοδιανύσματα προσθέτοντας επιπλέον όρους στα άκρα αυτών για να μεγαλώσει η πολλαπλότητα σε $p + 1$, έτσι ώστε να αποκτήσουμε τελικώς ανοιχτά κομβοδιανύσματα. Για παράδειγμα για $p = 3$ εάν το τοπικό κομβοδιάνυσμα είναι $U = \{0, 0, 1, 2, 3\}$ απλά προσθέτουμε τα $\{0, 0\}$ και $\{3, 3\}$ για να λάβουμε το επεκταμένο κομβοδιάνυσμα $\bar{U} = \{0, 0, 0, 0, 1, 2, 3, 3, 3, 3\}$. Επίσης ορίζουμε και την μεταβλητή n_t ως τον αριθμό των επιπλέων κόμβων που προστέθηκαν μπροστά (ή πίσω) από το κομβοδιάνυσμα και η αρίθμηση των συναρτήσεων βάσης T-splines απο αριστερά προς τα δεξιά θα ορίζεται ως $n_t + 1$.

Εφόσον έχουμε στα χέρια μας το επεκταμένο κομβοδιάνυσμα, ο τελεστής εξαγωγής Bézier υπολογίζεται όπως αναφέραμε και πριν όπως στην περίπτωση των NURBS, απλώς υπολογίζουμε μόνο την γραμμή που χρειαζόμαστε.

$$N_A(u_A)|_e = N_a^e(\bar{u}) = \mathbf{e}_a^T \mathbf{N}^e(\bar{u}) = \mathbf{e}_a^T \mathbf{C}^e \mathbf{B}(\bar{u}) = (\mathbf{c}_a^e)^T \mathbf{B}(\bar{u}) \tag{5.35}$$

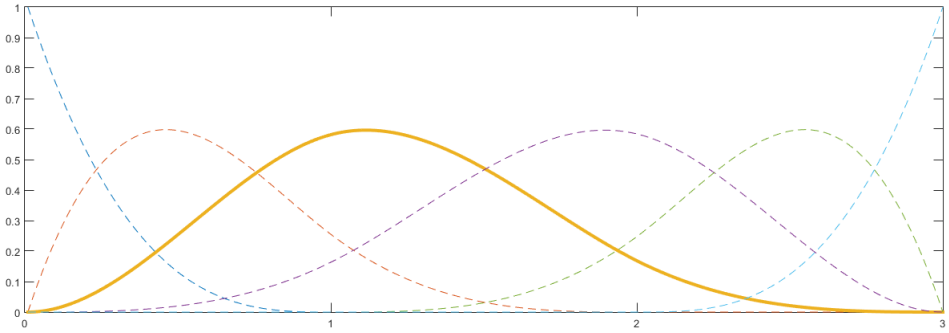
όπου το $\mathbf{e}_a$ είναι ένα μοναδιαίο διάνυσμα ίσο με 1 στην είσοδο a και 0 αλλού. Το διάνυσμα $\mathbf{c}_a^e$ εξάγει την συνάρτηση βάσης A για το στοιχείο e.

Τώρα για μία επιφάνεια χρειαζόμαστε διμερή εξαγωγή, που επιτυγχάνεται με το γινόμενο των συναρτήσεων βάσης σε κάθε παραμετρική κατεύθυνση. Η διμερής συνάρτηση βάσης λοιπόν θα ορίζεται ώς

$$N_A(\mathbf{u_A})|_e = N_a^e(\bar{\mathbf{u}}_A) = N_a^{e,1}(\bar{u}_1) N_a^{e,2}(\bar{u}_2) = \left[(\mathbf{c}_a^{e,1})^T \mathbf{B_1}(\bar{u}_1)\right]\left[(\mathbf{c}_a^{e,2})^T \mathbf{B_2}(\bar{u}_2)\right] \tag{5.36}$$

όπου με $1, 2$ συμβολίζουμε τις παραμετρικές κατευθύνσεις.

Συνοψίζοντας, οι έντονα χρωματισμένες γραμμές παρακάτω είναι αυτές που αφορούν τη συνάρτηση βάσης N_3. Στα σχήματα που ακολουθούν βλέπουμε της συναρτήσεις βάσης, τους τελεστές εξαγωγής Bézier και τα πολυώνυμα Bernstein μετά την εισαγωγή κόμβων.



Σχήμα 5.5: Η συνάρτηση βάσης N_3 της T-spline με έντονο κίτρινο χρώμα από το $U = \{0, 0, 1, 2, 3\}$. Οι υπόλοιπες συναρτήσεις βάσης απεικονίζονται με διακεκομμένες γραμμές

Σχήμα 5.6: Τα πολυώνυμα Bernstein για τα στοιχεία Bézier μετά την εισαγωγή κόμβων

$$\begin{Bmatrix} N_1 \\ N_2 \\ \mathbf{N_3} \\ N_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ \mathbf{0} & \mathbf{0} & \mathbf{1/2} & \mathbf{7/12} \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} \mathbf{B_1} \\ \mathbf{B_2} \\ \mathbf{B_3} \\ \mathbf{B_4} \end{Bmatrix} \tag{5.37}$$

$$\begin{Bmatrix} N_2 \\ \mathbf{N_3} \\ N_4 \\ N_5 \end{Bmatrix} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ \mathbf{7/12} & \mathbf{2/3} & \mathbf{1/3} & \mathbf{1/6} \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \begin{Bmatrix} \mathbf{B_4} \\ \mathbf{B_5} \\ \mathbf{B_6} \\ \mathbf{B_7} \end{Bmatrix} \tag{5.38}$$

$$\begin{Bmatrix} \mathbf{N_3} \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix} = \begin{bmatrix} \mathbf{1/6} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \mathbf{B_7} \\ \mathbf{B_8} \\ \mathbf{B_9} \\ \mathbf{B_{10}} \end{Bmatrix} \tag{5.39}$$

Για τον αριθμητικό υπολογισμό των μητρώων **C** που εμφανίζονται στις παραπάνω εξισώσεις, βλέπε Appendix A.

Κεφάλαιο 6

Ισογεωμετρική Ανάλυση

Η ισογεωμετρική ανάλυση χρησιμοποιεί το ίδιο μαθηματικό υπόβαθρο με την ανάλυση πεπερασμένων στοιχείων. Όμως, η ισογεωμετρική ανάλυση μας επιτρέπει να χρησιμοποιούμε τα εργαλεία του CAD για να αποτυπώσουμε την γεωμετρία του προβλήματος με απόλυτη ακρίβεια (όπου αυτό είναι δυνατόν). Σ' αυτό το κεφάλαιο λοιπόν θα αναφερθούμε στα πλεονεκτήματα της ισογεωμετρικής ανάλυσης αλλά και το πως συνδέεται μαθηματικά με τα καθιερωμένα πεπερασμένα στοιχεία. Τέλος, θα προσπαθήσουμε να προγραμματίσουμε σε Matlab έναν αλγόριθμο T-spline στις δύο διαστάσεις, που να ξεκινάει από ένα δοσμένο μη-δομημένο σύνολο κόμβων (knots) και εδρών (patches) και να δίνει στην έξοδο τα κομβοδιανύσματα, τις συναρτήσεις βάσης και τις παραγώγους τους. Έπειτα, θα επιλύσουμε ένα υπολογιστικό πρόβλημα μετάδοσης θερμότητας σε ένα ορθογώνιο χωρίο με βάση τα προηγούμενα.

6.1 Ισοπαραμετρική διακριτοποίηση

Η ισοπαραμετρική ιδέα αναφέρεται στην χρήση των ίδιων συναρτήσεων τόσο για την γεωμετρία όσο και για την διακριτοποίηση του άγνωστου πεδίου. Τόσο στην ισογεωμετρική ανάλυση όσο και στην ανάλυση πεπερασμένων στοιχείων οι μεθοδολογίες βασίζονται συνήθως στην παραμετρική ιδέα. Όμως διαφέρουν από τον τύπο συναρτήσεων βάσης που χρησιμοποιούν. Δηλαδή [8],

- Στα πεπερασμένα στοιχεία ή βάση που επιλέγεται για να προσδιορίσει το άγνωστο πεδίο είναι και η βάση που προσεγγίζει την γνωστή γεωμετρία. Αυτό συνήθως παίρνει την μορφή χαμηλής τάξης πολυωνύμικών συναρτήσεων και η γεωμετρία στις περισσότερες περιπτώσεις απλά προσεγγίζεται (π.χ. τόξο κύκλου).

- Στην ισογεωμετρική ανάλυση η βάση επιλέγεται για να αποτυπώσει πλήρως την γεωμετρία και χρησιμοποιείται για να προσεγγίσει το άγνωστο πεδίο. Βελτίωση μπορέι να χρειάζεται για τα άγνωστα πεδία, αλλά η ακριβής γεωμετρία διατηρείται σ' όλα τα στάδια της ανάλυσης.

Η μεθοδολογία του Galerkin μας επιτρέπει να γράψουμε την πλειοψηφία των προβλημάτων πεπερασμένων στοιχείων σε πολύ απλή μορφη. Αυτή η μεθοδολογία μπορεί να χρησιμοποιηθεί και στην ισογεωμετρική ανάλυση. Έτσι λοιπόν θα έχουμε

$$\mathbf{x} = \sum_{A=1}^{n} \Phi_A(\mathbf{u})\mathbf{P}_A \tag{6.1}$$

όπου Φ_A η συνάρτηση μορφής. Ο αριθμός των σημείων ελέγχου συμβολίζεται με n, $\mathbf{u}$ η παραμετρική συντεταγμένη και με $\mathbf{P}_A$ το σημείο ελέγχου αριθμού A. Σ' έναν ισοπαραμετρικό φορμαλισμό, η μεταβλητή του αγνώστου πεδίου προσεγγίζεται από την ίδια συνάρτηση μορφής όπως και στην γεωμετρία, δηλαδή

$$\mathbf{h}(x) = \sum_{A=1}^{n} \Phi_A(\mathbf{u})\mathbf{h}_A \tag{6.2}$$

όπου με $\mathbf{h}_A$ εννοούμε την τιμή της μεταβλητής που συσχετίζεται με το σημείο ελέγχου $\mathbf{P}_A$. Έτσι συνήθως αποκαλείται σαν μεταβλητή ελέγχου ή πιο γενικά ως βαθμός ελευθερίας. Συνεπώς τα παραπάνω μπορούν να γράψουν ένα πρόβλημα επίλυσης διαφορικών εξισώσεων στην απλή μορφή

$$\mathbf{Kh} = \mathbf{f} \tag{6.3}$$

όπου $\mathbf{K} = [\mathbf{K}]$ να ονομάζεται το *μητρώο στιβαρότητας (stiffness matrix)*, $\mathbf{h} = \{h\}$ και $\mathbf{f} = \{f\}$ διανύσματα στήλες. Όπως και στα συνηθισμένα πεπερασμένα στοιχεία ο πίνακας στιβαρότητας $[\mathbf{K}]$ και το διάνυσμα στήλη $\{f\}$ ή αλλιώς *διάνυσμα δύναμης (force vector)* μπορούν να κατασκευαστούν υπολογίζοντας ξεχωριστά το κάθε στοιχείο k^e, f^e.

6.2 Αριθμητική ολοκλήρωση

Αρχικά θα συμβολίσουμε με Ω τον φυσικό χώρο και με $\widehat{\Omega}$ τον παραμετρικό χώρο. Τα ολοκληρώματα υπολογίζονται στον κάθε φυσικό χώρο Ω^e που αντιστοιχεί στο κάθε στοιχείο e. Αυτά τα ολοκληρώματα μετά μετασχηματίζονται στον παραμετρικό χώρο $\widehat{\Omega}^e$ του κάθε στοιχείου μέσω του γεωμετρικού μετασχηματισμού. Τέλος, τα ολοκληρώματα καταλήγουν από τον παραμετρικό στον γονικό χώρο όπου και τα υπολογίζουμε κανονικά. Έτσι για μια συνάρτηση δύο μεταβλητών $f(x, y)$ θα έχουμε

$$\begin{aligned}\int_\Omega f(x,y)d\Omega &= \sum_{e=1}^{n}\int_{\Omega^e} f(x,y)d\widehat{\Omega}^e \\ &= \sum_{e=1}^{n}\int_{\widehat{\Omega}^e} f(x(u),y(v))|J_u|d\widehat{\Omega}^e \\ &= \sum_{e=1}^{n}\int_{\Gamma} f(\bar{u},\bar{v})|J_u|J_{\bar{u}}|d\Gamma\end{aligned} \tag{6.4}$$

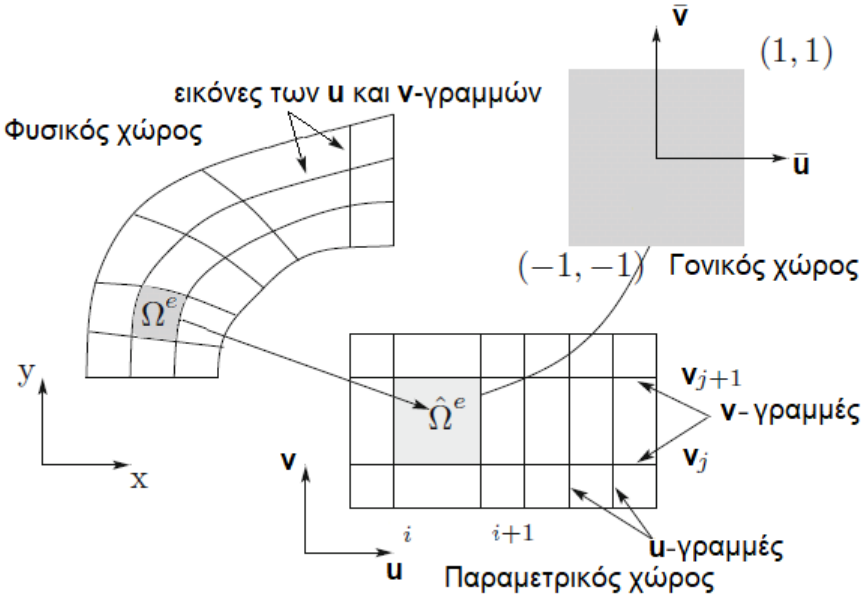
με Γ να είναι ο *γονικός χώρος (parent space)* και με $\mathbf{J_u}$ ο πίνακας Jacobi

$$\mathbf{J_u} = \begin{bmatrix} x_u & x_v \\ y_u & y_v \end{bmatrix} \tag{6.5}$$

Το τελικό ολοκλήρωμα μπορεί να υπολογιστεί με τον κανόνα ολοκλήρωσης Gauss. Ειδικότερα ένα $(p+1)\times(q+1)$ ολοκλήρωμα Gauss χρησιμοποιείται για δύο διαστάσεις με τα p, q να υποδηλώνουν τις τάξεις της βάσης NURBS στις κατευθύνσεις u, v. Παρακάτω στο Σχήμα 6.1 βλέπουμε τους χώρους ολοκλήρωσης που χρησιμοποιούνται στην ισογεωμετρική ανάλυση.

Ο μετασχηματισμός από το γονικό χώρο Γ στον παραμετρικό $\widehat{\Omega}^e = [u_i, u_{i+1}]\times[v_j, v_{j+1}]$ δίνεται από τις εξισώσεις

$$\begin{aligned} u &= \frac{1}{2}[(u_{i+1}-u_i)\bar{u} + (u_{i+1}+u_i)] \\ v &= \frac{1}{2}[(v_{i+1}-v_i)\bar{v} + (v_{i+1}+v_i)] \end{aligned} \tag{6.6}$$



Σχήμα 6.1: Χώροι ολοκλήρωσης στην ισογεωμετρική ανάλυση. Τα στοιχεία στον παραμετρικό χώρο ορίζονται σαν $\widehat{\Omega}^e = [u_i, u_{i+1}] \times [v_j, v_{j+1}]$ και τα στοιχεία του φυσικού χώρου είναι εικόνες των παραμετρικών [8].

όπου τα $\bar{u}, \bar{v}$ αναπαριστούν ένα σημείο Gauss. Έτσι, η ορίζουσα του Jacobian αυτού του μετασχηματισμού θα είναι

$$|J_{\bar{\mathbf{u}}}| = \frac{1}{4}(u_{i+1} - u_i)(v_{i+1} - v_i) \tag{6.7}$$

6.3 Αλγόριθμος T-splines

Για την κατασκευή ενός προγράμματος T-splines (στις 2 διαστάσεις) που να μας δίνει σαν έξοδο τα τοπικά κομβοδιανύσματα $\mathbf{U}_i, \mathbf{V}_i$ όλων των κόμβων (knots) του πλέγματος (T-mesh), και στη συνέχεια να τυπώνει τις συναρτήσεις βάσης και τις παραγώγους τους, ακολουθήθηκαν τα εξής βήματα (βλ. Appendix A):

- Ξεκινήσαμε εισάγοντας 6 παραδείγματα από τα οποία μπορεί να επιλέξει ο χρήστης όποιο επιθυμεί (γραμμή 6)
- Σαν επιπλέον είσοδο δώσαμε τον βαθμό πολυωνύμου p που μπορεί και πάλι να επιλέξει ο χρήστης (γραμμή 7)
- Υπολογίζουμε στην συνάρτηση calculateP το μήκος του κάθε τοπικού κομβοδιανύσματος σύμφωνα με το p που εισάγαμε (γραμμή 8)
- Ορίσαμε τα δεδομένα μας για κάθε παράδειγμα από τα 6. Τα δεδομένα που χρειαστήκαμε είναι τα σημεία (points) και οι έδρες (patches) (γραμμές 10-329)
- Υπολογίζουμε τα μοναδίκα οριζόντια και κάθετα κομβοδιανύσματα αφαιρώντας τυχόν όμοια ανά άξονα (γραμμές 335-344)
- Βρίσκουμε την ελάχιστη και την μέγιστη τιμή του κομβοδιανύσματος ανά άξονα (γραμμές 347-350)

- Με τη συνάρτηση detectAnomaly αναγνωρίζουμε πιθανόν τετριμμένη περίπτωση και αν όντως αναγνωριστεί τότε καλείται η συνάρτηση getAnomalyPoints με παραμέτρους points, patches και P (γραμμές 356-359)

- Στην περίπτωση μη τετριμμένης περίπτωσης καλούνται οι συναρτήσεις getPointsOnUAxis και getPointsOnVAxis οι οποίες επιστρέφουν τα οριζόντια και τα κάθετα τελικά κομβοδιανύσματα

- Αναλόγως την παράμετρο P οι παραπάνω δύο συναρτήσεις καλούνται με διαφορετικές παραμέτρους (γραμμή 364)

- Για περίπτωση περιττού P για κάθε point υπολογίζουμε κομβοδιάνυσμα παρέχοντας παραμέτρους points, points(x,:), P και αναλόγως άξονα Umin, Umax ή Vmin, Vmax (γραμμές 367-370)

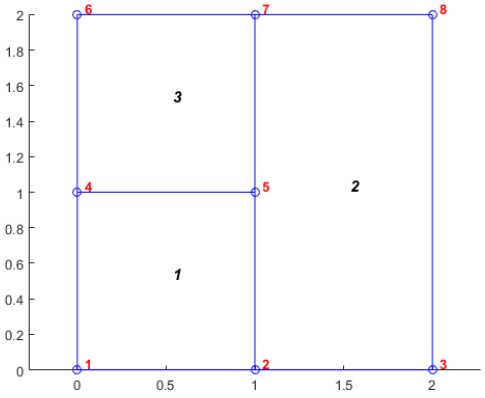
- Για περίπτωση άρτιου P για κάθε patch υπολογίζουμε το κέντρο center με τη συνάρτηση findRectangleCenter και παράμετρο patch(y,:). Έπειτα υπολογίζουμε κομβοδιάνυσμα παρέχοντας παραμέτρους points, center, P και αναλόγως άξονα Umin, Umax ή Vmin, Vmax (γραμμές 375-379)

- Τυπώνουμε τα τελικά κομβοδιανύσματα $\mathbf{U}_{vec}$ και $\mathbf{V}_{vec}$ για το παράδειγμα που επιλέξαμε κατα την έναρξη του προγράμματος (γραμμές 387-388)

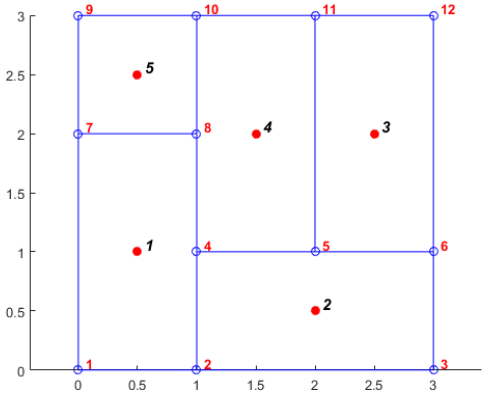
- Έπειτα ακολουθεί κώδικας ο οποίος σύμφωνα με τα τελικά $\mathbf{U}_{vec}$ και $\mathbf{V}_{vec}$ αναπαράγει και απεικονίζει σχηματικά τις συναρτήσεις βάσης Rip και τις παραγώγους τους derRip

Αξίζει να σημειωθεί πως ο χρήστης εάν το επιθυμεί μπορεί να αλλάξει και τις εντολές NsubdivisionsX, NsubdivisionsY στις γραμμές 456-459 που μας δίνουν τον αριθμό των υποδιαιρέσεων των γραφημάτων. Επίσης έαν το επιθυμεί ο χρήστης μπορέι να ελέγξει και την συνθήκη της διαμέρισης της μονάδος $\sum_{i=1}^{n} R_{i,p} = 1$ στις γραμμές 515-528.

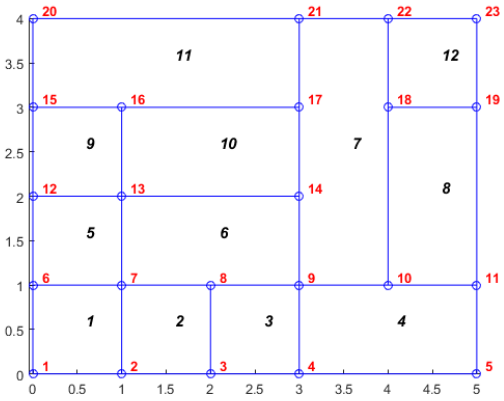
Παρακάτω στο Σχήμα 6.2 βλέπουμε τα meshes μαζί με την αρίθμηση τών κόμβων και των εδρών που χρησιμοποιήθηκαν και στα Σχήματα 6.3-6.8 τα γραφήματα των συναρτήσεων βάσης και των παραγώγων τους ανάλογα το παράδειγμα και το p που επιλέξαμε.



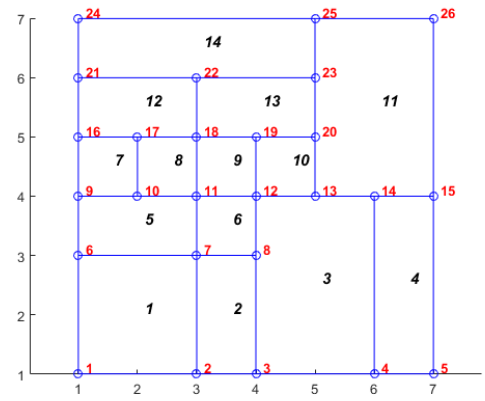
(α') Παράδειγμα 1



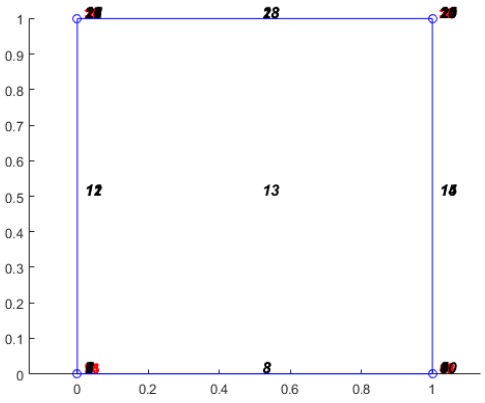
(β') Παράδειγμα 2



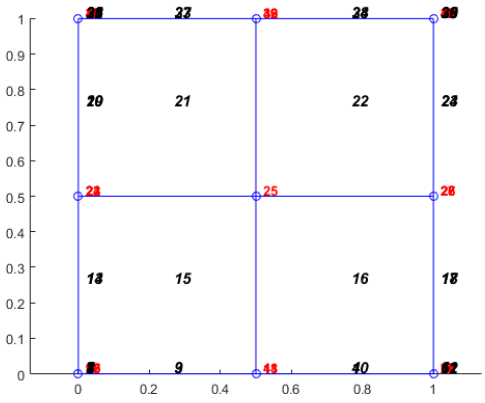
(γ') Παράδειγμα 3



(δ') Παράδειγμα 4

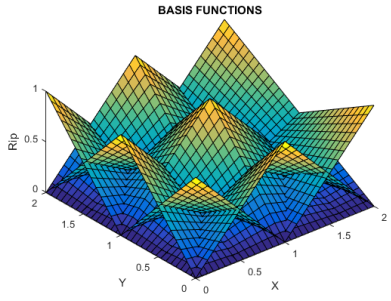


(ε') Παράδειγμα 5

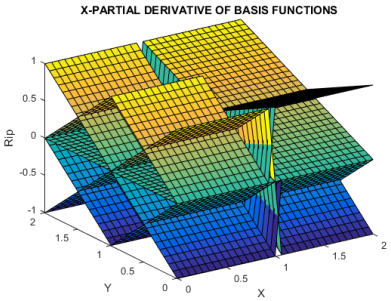


(ς') Παράδειγμα 6

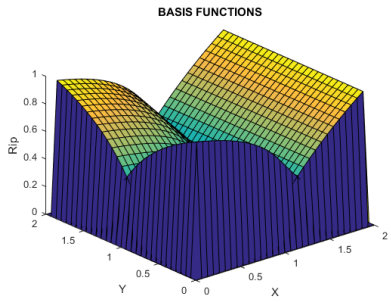
Σχήμα 6.2: T-meshes παραδείγματα 1-6 που χρησιμοποιήθηκαν



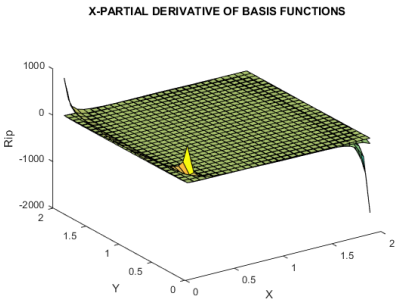
(α΄) $p = 1$



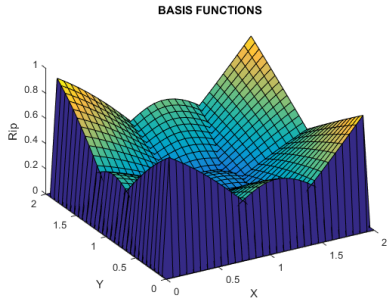
(β΄) $p = 1$



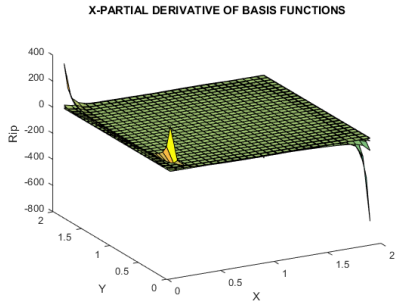
(γ΄) $p = 2$



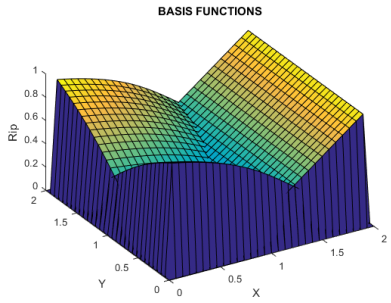
(δ΄) $p = 2$



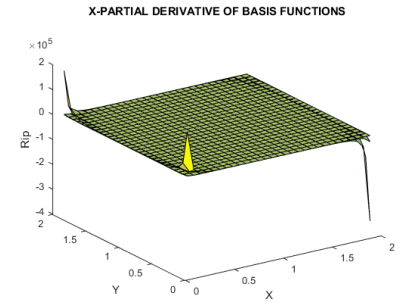
(ε΄) $p = 3$



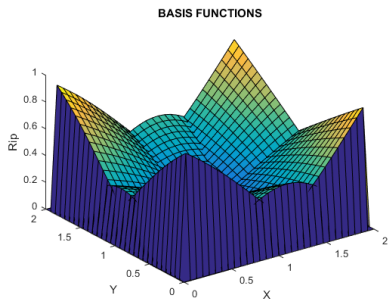
(ϛ΄) $p = 3$



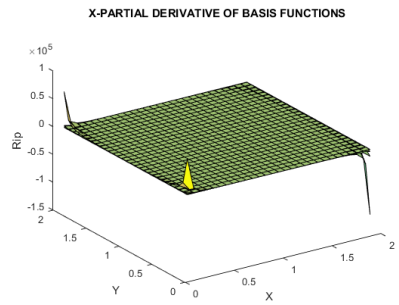
(ζ) $p = 4$



(η΄) $p = 4$

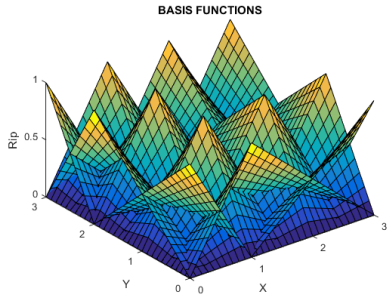


(θ΄) $p = 5$

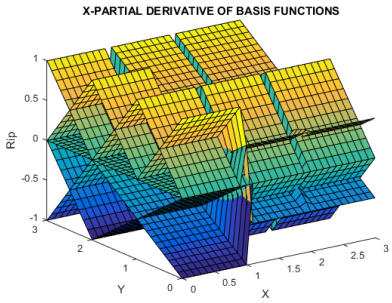


(ι΄) $p = 5$

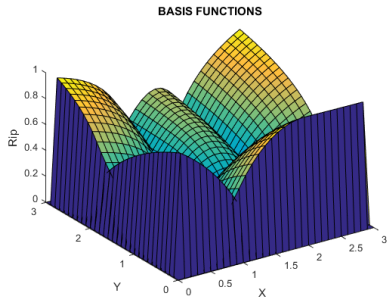
Σχήμα 6.3: Συναρτήσεις βάσης και παράγωγοι για το παράδειγμα 1



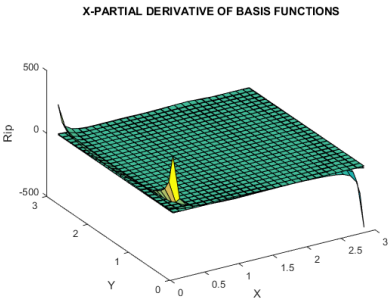
(α΄) $p = 1$



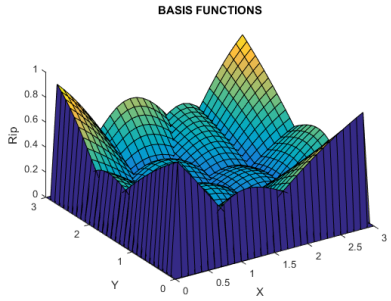
(β΄) $p = 1$



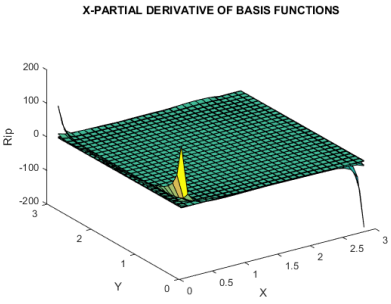
(γ΄) $p = 2$



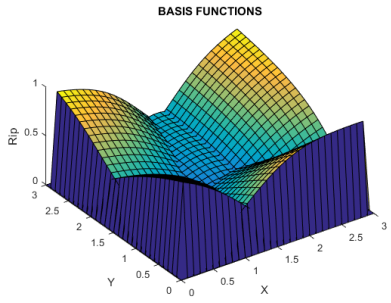
(δ΄) $p = 2$



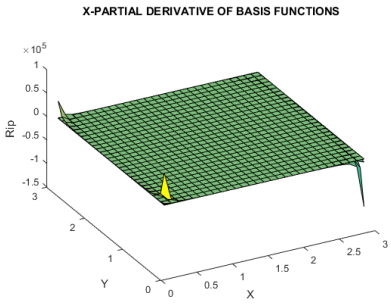
(ε΄) $p = 3$



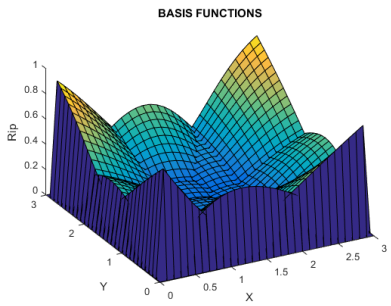
(ϛ΄) $p = 3$



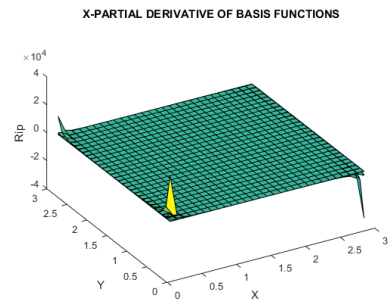
(ζ) $p = 4$



(η΄) $p = 4$

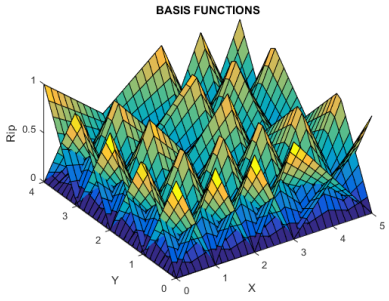


(θ΄) $p = 5$

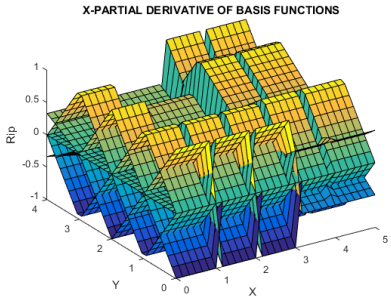


(ι΄) $p = 5$

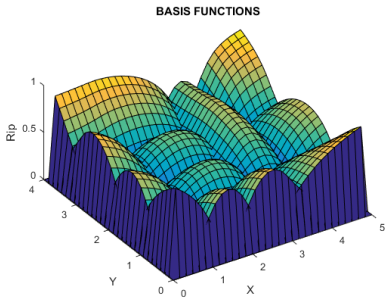
Σχήμα 6.4: Συναρτήσεις βάσης και παράγωγοι για το παράδειγμα 2



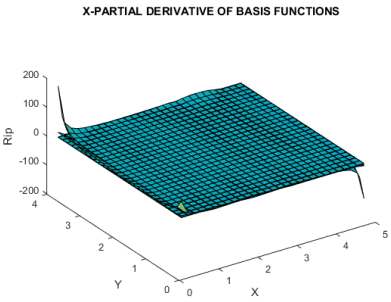
(α΄) $p = 1$



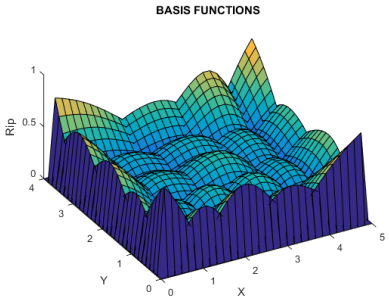
(β΄) $p = 1$



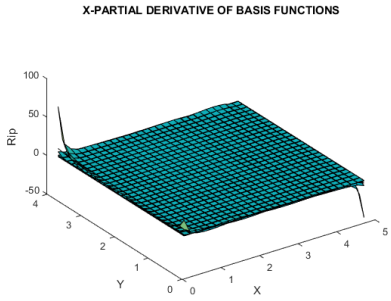
(γ΄) $p = 2$



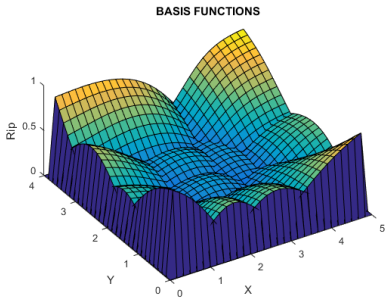
(δ΄) $p = 2$



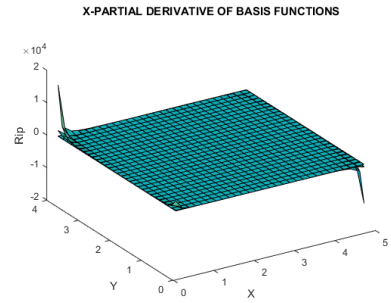
(ε΄) $p = 3$



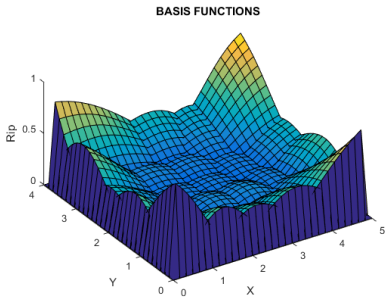
(ϛ΄) $p = 3$



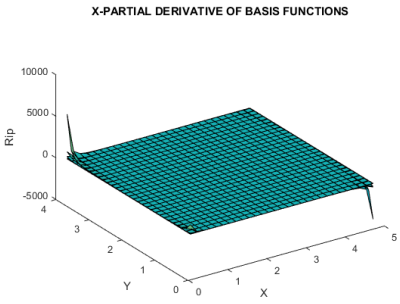
(ζ) $p = 4$



(η΄) $p = 4$

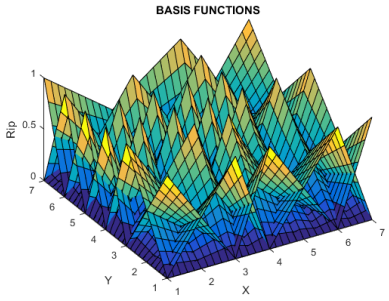


(θ΄) $p = 5$

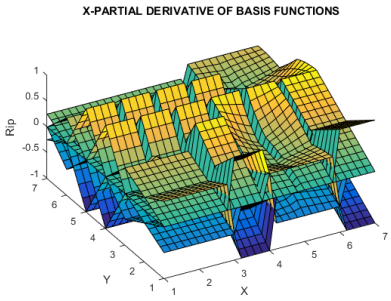


(ι΄) $p = 5$

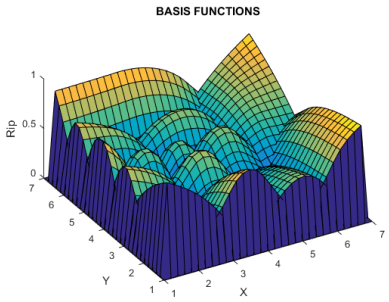
Σχήμα 6.5: Συναρτήσεις βάσης και παράγωγοι για το παράδειγμα 3



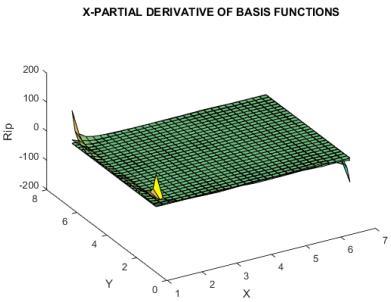
(α΄) $p = 1$



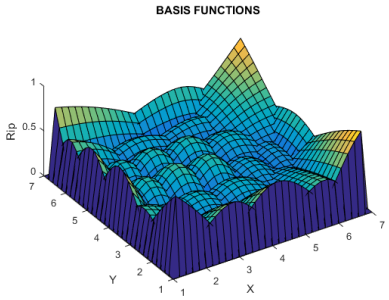
(β΄) $p = 1$



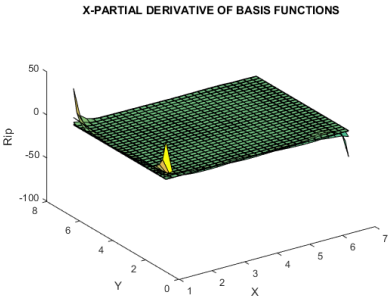
(γ΄) $p = 2$



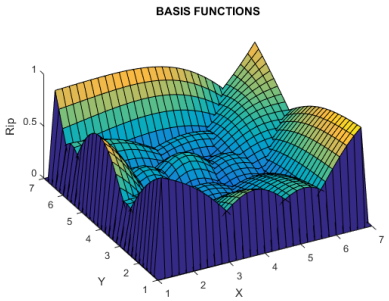
(δ΄) $p = 2$



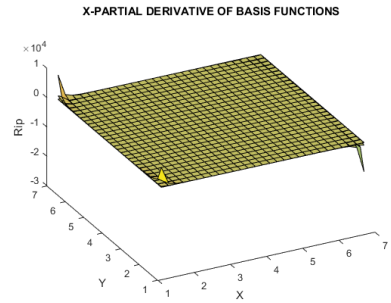
(ε΄) $p = 3$



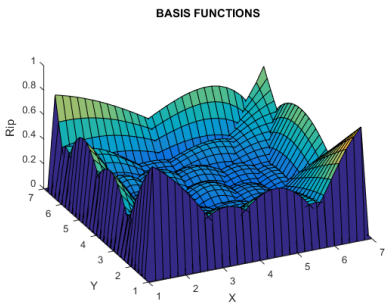
(ϛ΄) $p = 3$



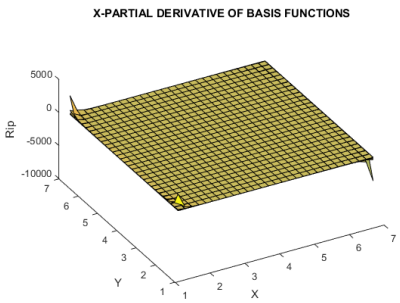
(ζ) $p = 4$



(η΄) $p = 4$

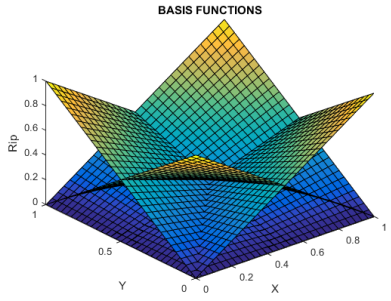


(θ΄) $p = 5$

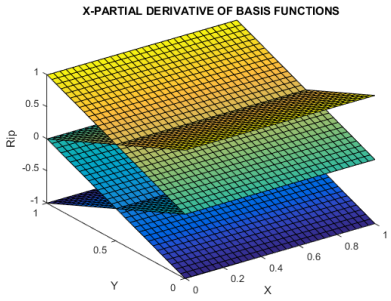


(ι΄) $p = 5$

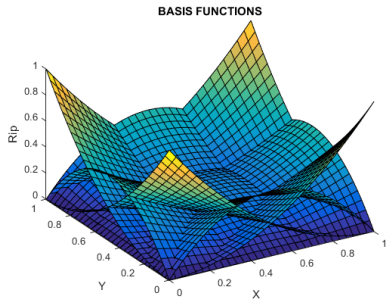
Σχήμα 6.6: Συναρτήσεις βάσης και παράγωγοι για το παράδειγμα 4



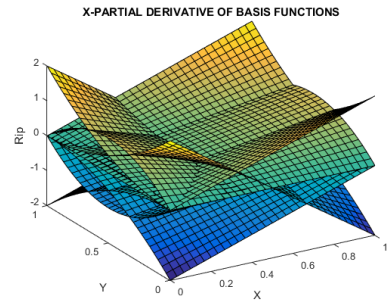
(α΄) $p = 1$



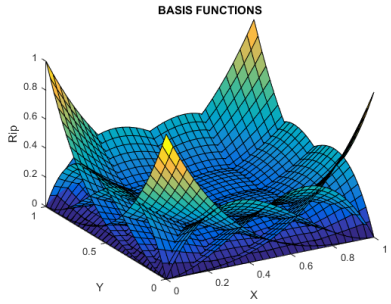
(β΄) $p = 1$



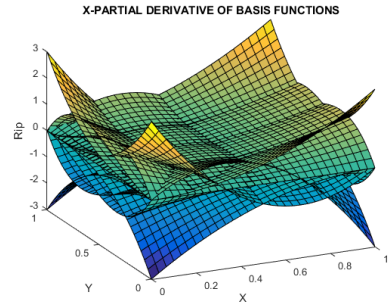
(γ΄) $p = 2$



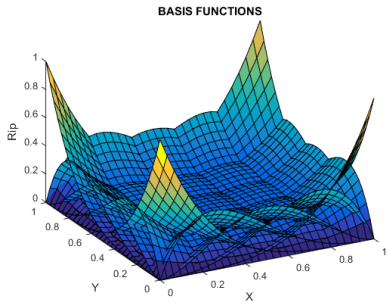
(δ΄) $p = 2$



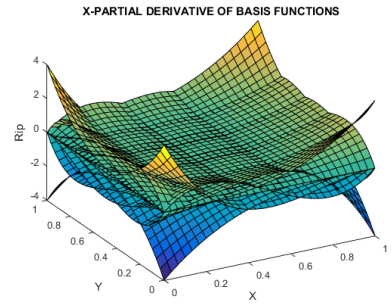
(ε΄) $p = 3$



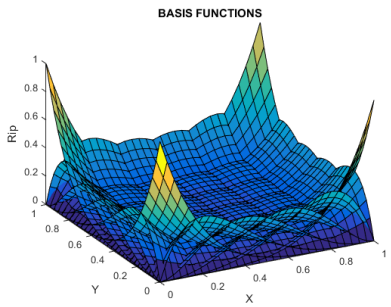
(ϛ΄) $p = 3$



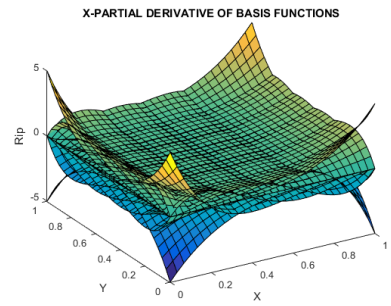
(ζ) $p = 4$



(η΄) $p = 4$

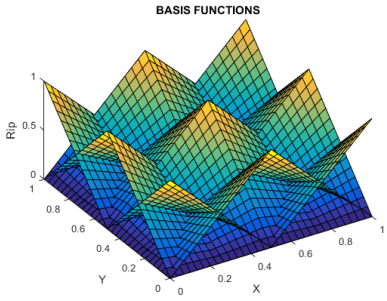


(θ΄) $p = 5$

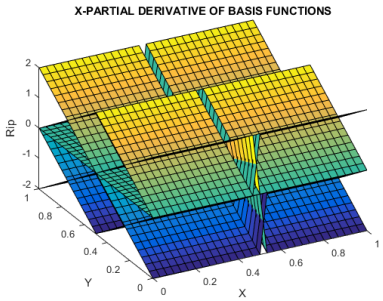


(ι΄) $p = 5$

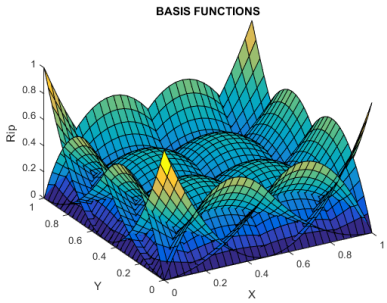
Σχήμα 6.7: Συναρτήσεις βάσης και παράγωγοι για το παράδειγμα 5



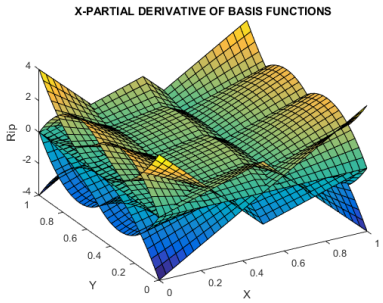
(α΄) $p = 1$



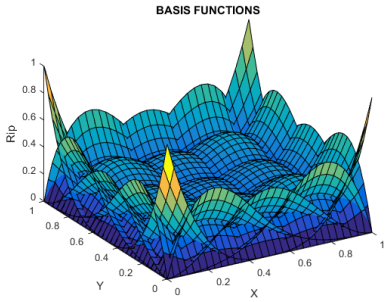
(β΄) $p = 1$



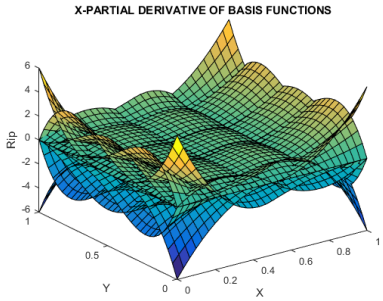
(γ΄) $p = 2$



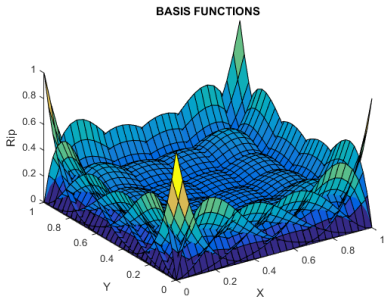
(δ΄) $p = 2$



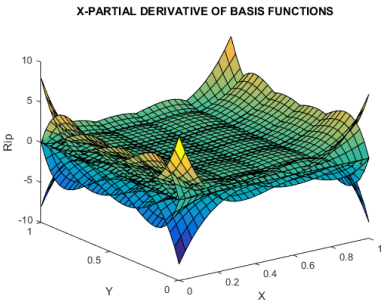
(ε΄) $p = 3$



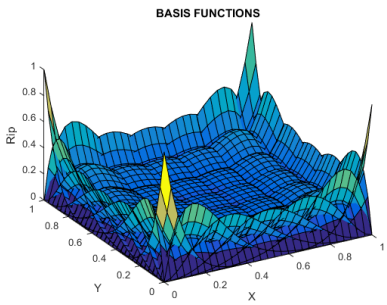
(ϛ΄) $p = 3$



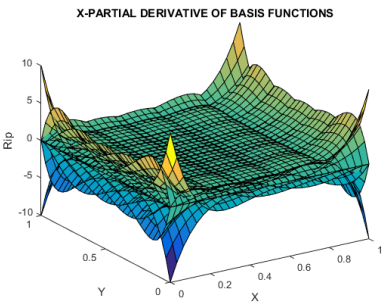
(ζ) $p = 4$



(η΄) $p = 4$



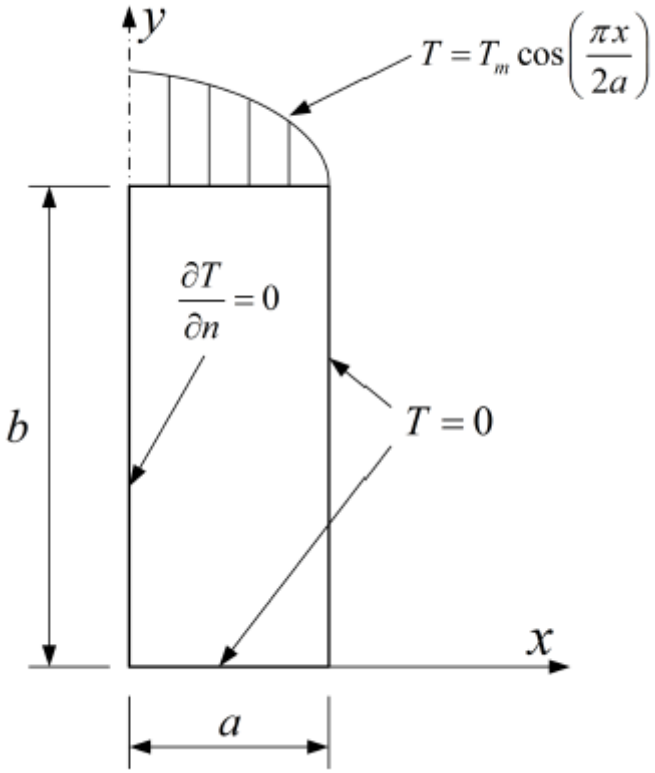
(θ΄) $p = 5$



(ι΄) $p = 5$

Σχήμα 6.8: Συναρτήσεις βάσης και παράγωγοι για το παράδειγμα 6

6.4 Υπολογιστική εφαρμογή



Σχήμα 6.9: Ορθογώνιο χωρίο

Οι προηγούμενες συναρτήσεις βάσης θα εφαρμοσθούν για την επίλυση της εξίσωσης Laplace ($\nabla^2 T = 0$) που περιγράφει τη μόνιμη ροή θερμότητας σε ένα ορθογώνιο χωρίο διαστάσεων $a \times b = 3 \times 12$ για τις οριακές συνθήκες (εν μέρει τύπου Dirichlet, εν μέρει Neumann) που φαίνονται στο Σχήμα 6.9.

Για κάθε σημείο (x, y) του πεδίου ορισμού ($\Omega = [0, a] \times [0, b]$) , η θεωρητική λύση του προβλήματος δίνεται από τη σχέση

$$T(x, y) = T_m \left(\frac{sinh(\frac{\pi y}{2a})}{sinh(\frac{\pi b}{2a})} \right) cos(\frac{\pi x}{2a}) \tag{6.8}$$

Η μέγιστη θερμοκρασία ελήφθη αυθαίρετα ίση προς $T_m = 100^o C$, τόσο για τη συμβατική μέθοδο των πεπερασμένων στοιχείων όσο και για την ισογεωμετρική ανάλυση, το μητρώο στιβαρότητας εκφράζεται συναρτήσει των διδιάστατων συναρτήσεων βάσης, $N_i(x, y)$, και δίνεται διαδοχικά από τις σχέσεις (βλ. π.χ. [19])

$$\begin{aligned} k_{ij} &= \int_\Omega \nabla N_i \nabla N_j d\Omega \\ &= \int_\Omega \left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} \right) d\Omega \end{aligned} \tag{6.9}$$

Στην περίπτωση της ισογεωμετρικής ανάλυσης, κάθε συνάρτηση βάσης N_i επιδρά καταρχήν σε ¨ολόκληρο¨ το χωρίο. Ωστόσο, η εγγενής των B-splines *συμπαγής υποστήριξη (compact support)* περιορίζει τη δράση της N_i , έτσι ώστε να είναι μη-μηδενική

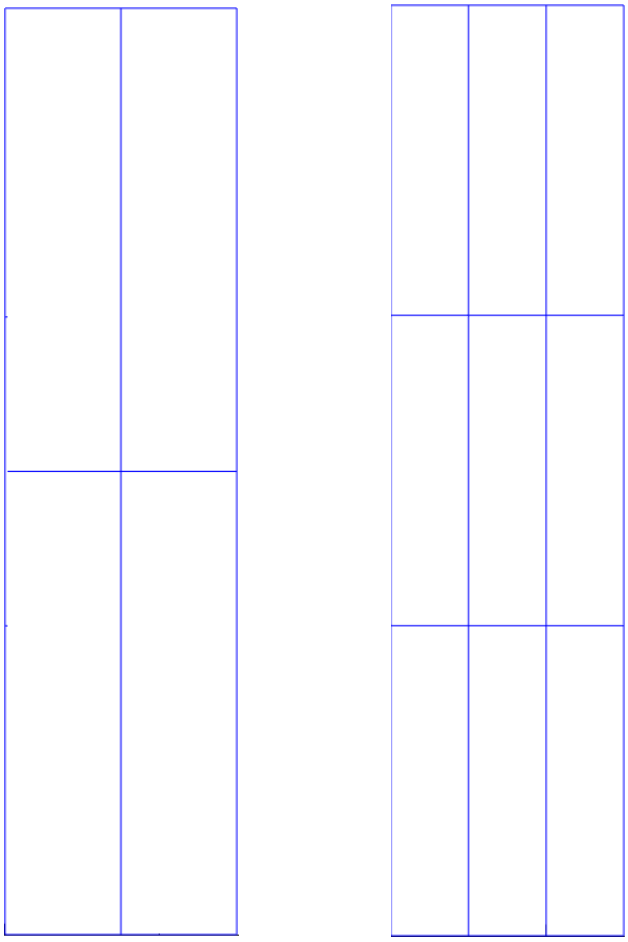
μόνο εντός του υποδιαστήματος $[u_i, u_{i+p+1})$ στην κατεύθυνση x , και του υποδιαστήματος στην $[v_i, v_{i+p+1})$ κατεύθυνση y (κανόνας de Boor, 1972). Αυτό σημαίνει ότι στα εν λόγω επιφανειακά knot spans (π.χ. $[u_i, u_{i+1}] \times [v_i, v_{i+1}]$, κλπ.) πρέπει να επιτελέσουμε ολοκλήρωση Gauss ($n_g \times n_g$), καθόσον ο πολυωνυμικός βαθμός p παραμένει σταθερός ανά κατεύθυνση. Αντίθετα, στην περίπτωση των NURBS η ρητή διατύπωσή τους αποκλίνει από το τέλειο πολυώνυμο και τότε απαιτείται είτε ειδική αριθμητική ολοκλήρωση ή περισσότερα σημεία Gauss από το κανονικό ($2n_g - 1 \geq p$).

Σημειώνουμε ότι στην ειδική περίπτωση των συναρτήσεων Bézier-Bernstein, εάν αυτές επιλεγούν να εφαρμόζονται σε ολόκληρο το πεδίο ορισμού (υπό τη μορφή ενός μοναδικού μακροστοιχείου), τότε απαιτείται ένα μόνο κελί ολοκλήρωσης που είναι ταυτόσημο με το πεδίο διαστάσεων $a \times b = 3 \times 12$.

Το παράδειγμα αυτό επιλύεται για τις ακόλουθες περιπτώσεις:

- Πολυώνυμα Bézier-Bernstein με ομοιόμορφο πολυωνυμικό βαθμό: $p = 3, 4, 5, 6, 7, 8, 9$ και στις δύο κατευθύνσεις για απλό ορθογώνιο χωρίς εσωτερικού κόμβους σαν το Παράδειγμα 5 που είδαμε πιο πριν (σελ. 38).

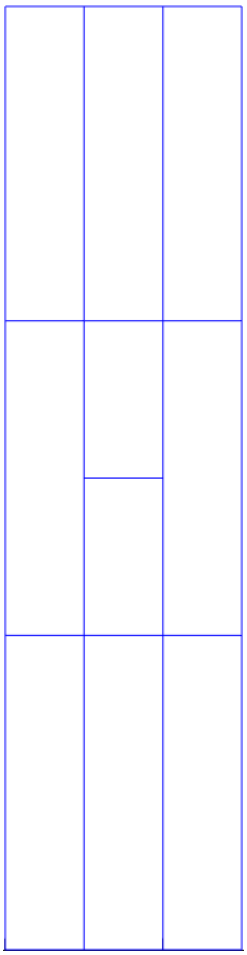
- B-splines για $p = 3, 4, 5, 6, 7, 8, 9$ με εσωτερικούς κόμβους ομοιόμορφα μοιρασμένους στα σημεία: $(x = a/2)$ ή $(x = a/3 \ \ \& \ \ x = 2a/3)$ και ομοίως στη κατεύθυνση y για τα ορθογώνια πλέγματα στο Σχήμα 6.10.



(α') Παράδειγμα 6 (β') Παράδειγμα 7

Σχήμα 6.10: Ορθογώνια πλέγματα με εσωτερικούς κόμβους (α) $x = a/2$, $y = b/2$, (β) $x = a/3$, $x = 2a/3$ και $y = b/3$, $y = 2b/3$

- Κυβικό T-spline για την τοπολογία που φαίνεται στο Σχήμα 6.11.



Σχήμα 6.11: Παράδειγμα 8 ενός ορθογώνιου πλέγματος T-mesh

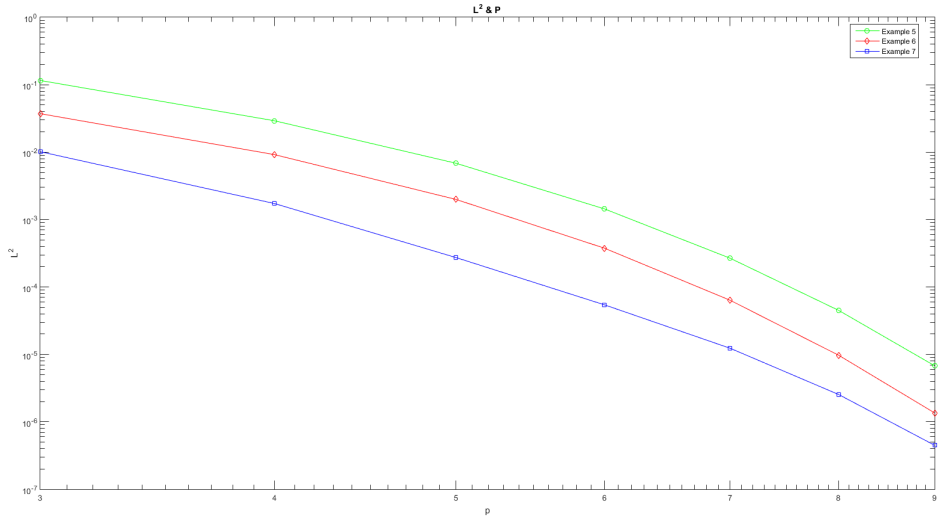
Σε όλες τις περιπτώσεις, η νόρμα L_2 του σφάλματος δίνεται από τη σχέση

$$L_2 = \frac{\sqrt{\int_\Omega (T - T_{exact})^2 dT}}{\int_\Omega (T_{exact}^2) dT} \tag{6.10}$$

T_{exact} είναι η ακριβής λύση που δίνεται από τη σχέση (6.8), και T είναι η υπολογισθείσα προσεγγιστική λύση με την οποιαδήποτε μέθοδο IGA.

Για την πρώτη και δεύτερη περίπτωση των πολυωνύμων Bézier-Bernstein (Παράδειγμα 5) και των B-splines (Παράδειγμα 6 και 7) αντίστοιχα βλέπουμε παρακάτω στο Σχήμα 6.12 σε διπλή λογαριθμική κλίμακα με ορίζοντιο άξονα τον πολυωνυμικό βαθμό p και κάθετο την ${L_2}^{(i)}$ νόρμα που μας δίνει μονοτονική μείωση με $(i) = 0, 1, 2$ να συμβολίζουμε τον αριθμό των εσωτερικών κόμβων για το κάθε πλέγμα.

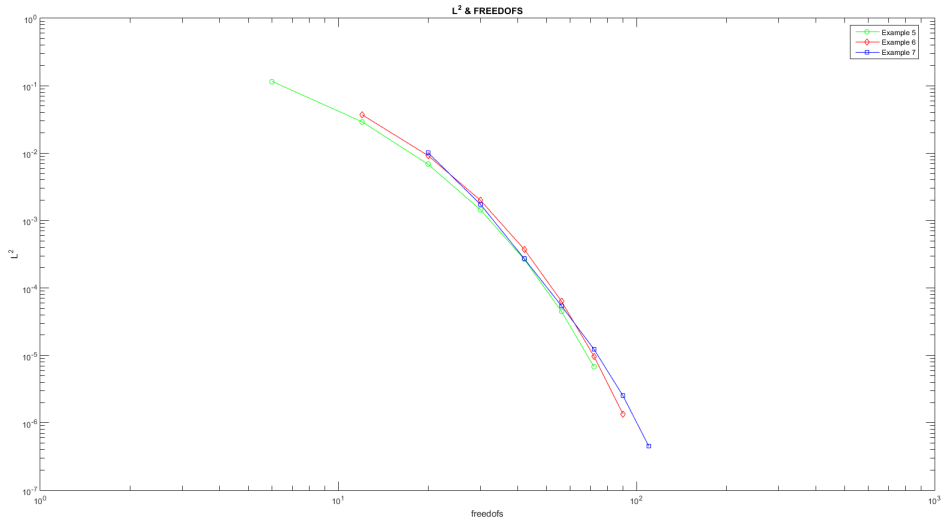
p	Bézier ${L_2}^{(0)}$	B-splines ${L_2}^{(1)}$	B-splines ${L_2}^{(2)}$
3	$1.1557383e-01$	$3.7211114e-02$	$1.0153930e-02$
4	$2.9200782e-02$	$9.1894482e-03$	$1.7294518e-03$
5	$6.8337974e-03$	$1.9872595e-03$	$2.7286905e-04$
6	$1.4379160e-03$	$3.7582399e-04$	$5.4323263e-05$
7	$2.6748327e-04$	$6.3678368e-05$	$1.2358215e-05$
8	$4.4921638e-05$	$9.7054557e-06$	$2.5400716e-06$
9	$6.7942355e-06$	$1.3562341e-06$	$4.4912309e-07$



Σχήμα 6.12: Διάγραμμα λογαριθμικής κλίμακας L_2 και p

Επίσης σε διπλή λογαριθμική κλίμακα μπορούμε να δούμε το διαγραμμα στο Σχήμα 6.13 των βαθμών ελευθερίας freedofs που συμμετέχουν στο πρόβλημα σε σχέση και πάλι με τις νόρμες ${L_2}^{(i)}$ (απότομη μείωση).

DOF	Bézier ${L_2}^{(0)}$	B-splines ${L_2}^{(1)}$	B-splines ${L_2}^{(2)}$
6 |12 |20	$1.1557383e-01$	$3.7211114e-02$	$1.0153930e-02$
12 |20 |30	$2.9200782e-02$	$9.1894482e-03$	$1.7294518e-03$
20 |30 |42	$6.8337974e-03$	$1.9872595e-03$	$2.7286905e-04$
30 |42 |56	$1.4379160e-03$	$3.7582399e-04$	$5.4323263e-05$
44 |56 |72	$2.6748327e-04$	$6.3678368e-05$	$1.2358215e-05$
56 |72 |90	$4.4921638e-05$	$9.7054557e-06$	$2.5400716e-06$
72 |90 |110	$6.7942355e-06$	$1.3562341e-06$	$4.4912309e-07$



Σχήμα 6.13: Διάγραμμα λογαριθμικής κλίμακας L_2 και freedofs

Για την τελευταία περίπτωση των T-splines (Παράδειγμα 8) συγκρίνοντάς την με αυτή των B-splines για $p = 3$ παρατηρούμε ότι δεν υπάρχει ιδαίτερη διαφορά στο σφάλμα παρά την μικρή πύκνωση του πλέγματος στην περίπτωση των T-splines. Αναφορικά στις B-splines για το πλέγμα με τους δύο εσωτερικούς κόμβους είχαμε βρει ότι ${L_2}^{(2)} = 1.0153930e-02$ ενώ για τις T-splines βρήκαμε ${L_2}^{(T)} = 9.9802392e-03$ που ουσιαστικά δεν διαφέρει ιδιαίτερα από την ${L_2}^{(2)}$ για $p = 3$ καθώς η διαφορά τους είναι μόλις 0.0001736908. Προφάνως θα ισχύουν τα ανάλογα για διάφορα p.

Αξίζει σ' αυτό το σημείο να αναφερθούμε στην μετατροπή του κύριου αλγορίθμου της έυρεσης των U_{vec}, V_{vec} που χρειάστηκε για το T-mesh του Παραδείγματος 8 (βλ. Appendix A).

- Η συνάρτηση getData (γραμμή 12) μας επιστρέφει τον πίνακα σημείων σε δύο μορφές κάθως και τον πίνακα εδρών.

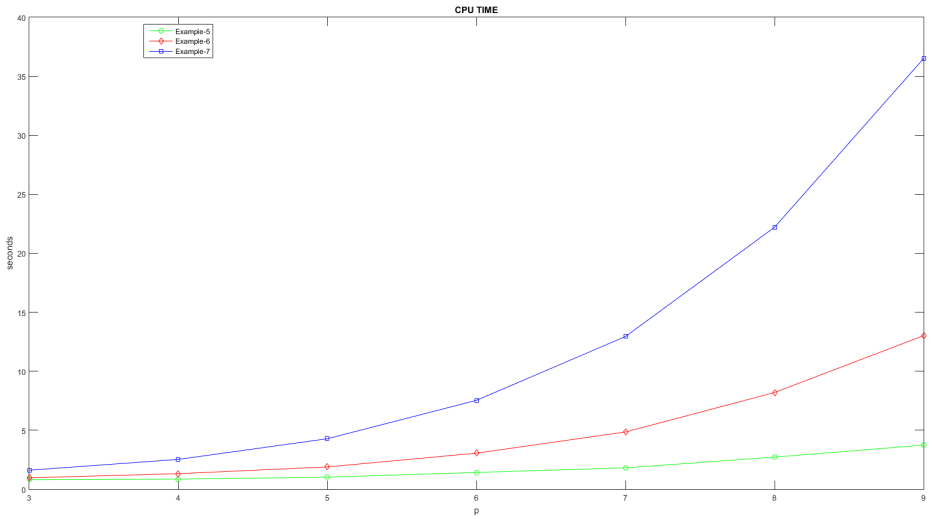
- Επίσης δημιουργήσαμε δύο νέες συναρτήσεις. Αυτές είναι η calculateEvenMesh (γραμμή 42) όπου υπολογίζει τα U_{vec}, V_{vec} για άρτιο p και η calculateOddMesh (γραμμή 49) όπου υπολογίζει τα U_{vec}, V_{vec} για περιττό p.

- Η συνάρτηση calculateOddMesh παίρνοντας σαν παραμέτρους τον πίνακα σημείων σε μορφή όπου καθορίζεται η έλλειψη διασταυρώσεων δημιουργεί δύο διανύσματα, ένα για U και ενα για V. Έπειτα υπολογίζει τον αριθμό των στοιχείων του διανύσματος για καθε πλευρά και μετά από μερικές επαναλήψεις συμπληρώνει το διάνυσμα όπως αυτό χρειάζεται. Σημαντικό ρόλο στη λύση αυτής της περίπτωσης έπαιξε ο τρόπος δήλωσης των στοιχείων.

- Με την ίδια λογική λειτουργεί και η συνάρτηση calculateEvenMesh με μόνη διάφορα οτι εδώ παίρνουμε σαν παράμετρο τον πίνακα των εδρών. Επίσης υπολογίζουμε τα *max*, *min* για τα U, V της κάθε έδρας με αποτέλεσμα να έχουμε τα πρώτα στοιχεία του τελικού διανύσματος για κάθε σημείο. Πέραν αυτού, υπολογίζουμε τον αριθμό των στοιχείων που απομένουν για κάθε πλευρά του διανύσματος και μετά από μερικές επαναλήψεις έχουμε το επιθυμητό αποτέλεσμα.

Όπως είναι λογικό, ο χρόνος που θα χρειαστεί ο ηλεκτρονικός υπολογιστής για να εκτελέσει το πρόγραμμα εξαρτάται από τον πολυωνυμικό βαθμό p και την πύκνωση του πλέγματος. Αυξάνοντας ένα από τα δύο ή και τα δύο παρατηρούμε πως στο παρακάτω Σχήμα 6.14 ιδιαίτερα για την περίπτωση του πλέγματος των δύο εσωτερικών κόμβων πως η αύξηση αρχίζει και γίνεται εκθετική. Οι στήλες CPU1 Time, CPU2 Time και CPU3 Time ανιστοιχούν στο απλό οθρογώνιο, το ορθογώνιο με ένα εσωτερικό κόμβο στο μέσο του διαστήματος και το ορθογώνιο με τους δύο εσωτερικούς κόμβους στα 1/3 και 2/3 των αξόνων για τα Παραδείγματα 5, 6 και 7.

p	CPU1 Time	CPU2 Time	CPU3 Time
3	$0.827s$	$0.969s$	$1.616s$
4	$0.860s$	$1.321s$	$2.530s$
5	$1.024s$	$1.893s$	$4.284s$
6	$1.426s$	$3.059s$	$7.535s$
7	$1.812s$	$4.856s$	$12.947s$
8	$2.725s$	$8.200s$	$22.204s$
9	$3.743s$	$13.019s$	$36.504s$



Σχήμα 6.14: Διάγραμμα CPU time και p

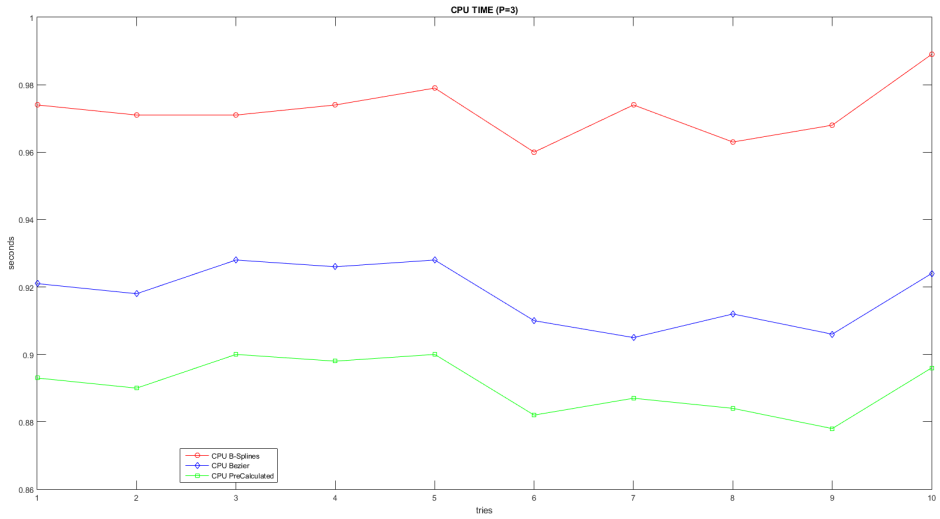
Έπειτα εφαρμόσαμε την εξαγωγή Bézier για $p = 3$ στο ορθογώνιο πλέγμα με τους δύο εσωτερικούς κόμβους και την συγκρίναμε με την συμβατική μέθοδο των B-splines. Παρατηρήσαμε ότι η εξαγωγή Bézier (CPU Bézier) υπερτερεί σημαντικά έναντι της συμβατικής μεθόδου των B-splines (CPU Simple) ιδιαίτερα εάν έχουμε υπολογίσει από πριν και τα πολυώνυμα των Bézier-Bernstein (CPU Bézier PreCalc) (βλ. Appendix A).

Tries	CPU Simple	CPU Bézier	CPU Bézier PreCalc
1	$0.974s$	$0.921s$	$0.893s$
2	$0.971s$	$0.918s$	$0.890s$
3	$0.971s$	$0.928s$	$0.900s$
4	$0.974s$	$0.926s$	$0.898s$
5	$0.979s$	$0.928s$	$0.900s$
6	$0.960s$	$0.910s$	$0.882s$
7	$0.974s$	$0.905s$	$0.887s$
8	$0.963s$	$0.912s$	$0.884s$
9	$0.968s$	$0.906s$	$0.878s$
10	$0.989s$	$0.924s$	$0.896s$

Οι ηλεκτρονικοί υπολογιστές όταν εκτελούν μία ορισμένη διαδικασία αρκετές φορές και στην περίπτωση μας αλγορίθμους, αν και δεν πέφτει στην αντίληψή μας πάντα έχουν μία μικρή διακύμανση στο χρόνο εκτέλεσής της. Αυτή η *διακύμανση* S για την διακριτή μεταβλητή x και την μέση τιμή μ για σύνολο παρατηρήσεων $n = 10$ δίνεται από τη σχέση

$$S = \sqrt{\left(\frac{\sum_{i=1}^{n}(x-\mu)^2}{n}\right)} \tag{6.11}$$

όπου παρατηρήσαμε πως οι διακυμάνσεις για τις μεθόδους είναι $S_1 = 0.0077$, $S_2 = 0.0085$ και $S_3 = 0.0087$ για την συμβατική μέθοδο των B-splines, την εξαγωγή Bézier και την εξαγωγή Bézier με προϋπολογισμένα τα πολυώνυμα Bernstein αντίστοιχα. Σύμφωνα με αυτές τις διακυμάνσεις δεν υπάρχει περίπτωση ποτέ να νικήσουν οι συμβατικές μέθοδοι τον χρόνο υπολογισμού της εξαγωγής Bézier.



Σχήμα 6.15: Διάγραμμα CPU time με τον αριθμό προσπαθειών για τις μεθόδους B-splines και Bézier

Αξίζει να σημειώσουμε πως οι παραπάνω χρόνοι υπολογισμού εξαρτώνται από το πόσο καλό *hardware* και *software* διαθέτει ο εκάστοτε ηλεκτρονικός υπολογιστής που χρησιμοποιούμε. Στην δική μας περίπτωση τα τεχνικά χαρακτηριστικά του ηλεκτρονικού μας υπολογιστή ήταν:

- CPU: Intel Core i7-7700k @ 4.7Ghz
- RAM: 16GB DDR4 3333Mhz CL16
- GPU: Asus GeForce GTX ROG Strix 1080Ti 11GB
- SSD: Samsung 256 850 Pro Series
- OS: Windows 10 Home 64-bit, Version 1909

Τέλος οι εφαρμογές όπως π.χ. το προγραμματιστικό περιβάλλον Matlab χρησιμοποιούν κυρίως την CPU και την RAM. Οι ταχύτητες αυτές θα ήταν σαφώς μικρότερες εάν αξιοποιούσαμε την GPU του υπολογιστή αντί της CPU οι οποία υπερτερεί, καθώς διαθέτει μεγαλύτερο σετ υπολογιστικών μεθόδων και είναι εξειδικευμένη σε πράξεις μεταξύ πινάκων και διανυσμάτων. Αυτή την στιγμή το Matlab υποστηρίζει μόνο κάρτες γραφικών NVIDIA για γρηγορότερους υπολογισμούς.

Κεφάλαιο 7

Επίλογος

Κύριος στόχος αυτής της μεταπτυχιακής εργασίας ήταν η κατασκευή από την αρχή, ενός αλγορίθμου ο οποίος να μπορεί να διαχειρίζεται οποιοδήποτε ορθογώνιο πλέγμα (T-mesh) του δοθεί με μόνη είσοδο τα σημεία και τις έδρες του εκάστοτε πλέγματος. Με βάση αυτά, παρήχθησαν τα τοπικά κομβοδιανύσματα του κάθε σημείου για να χρησιμοποιηθούν στη συνέχεια σε υπολογισμούς όπως η κατασκευή των συναρτήσεων βάσης και των γραφημάτων τους αλλά και η εύρεση της L_2 νόρμας στην περίπτωση μας για το πρόβλημα της μετάδοσης θερμότητας (εξίσωση Laplace).

Επίσης δείξαμε την υπεροχή της εξαγωγής Bézier έναντι των συμβατικών μεθόδων. Πιο συγκεκριμένα για πολυωνυμικό βαθμό $p = 3$ καταφέραμε και εφαρμόσαμε ένα απλό παράδειγμα της εξαγωγής Bézier για τις B-splines στο οθρογώνιο πλέγμα με τους δύο εσωτερικούς κόμβους δείχνοντας έτσι πως υπερτερεί στο χρόνο εκτέλεσης σε σχέση με την συμβατική μέθοδο των B-splines.

Σε περιπτώσεις με μεγάλους πολυωνυμικούς βαθμούς p και μεγάλο πλήθος στοιχείων n θα ήταν θεμιτό να υπολογίσουμε από πριν τους τελεστές εξαγωγής C. Παρ' όλα αυτά εάν η αύξηση των δύο παραπάνω παραμέτρων είναι υπερβολική μπορεί να μας οδηγήσει σε προβλήματα έλλειψης διαθέσιμης RAM με αποτέλεσμα ο ηλεκτρονικός υπολογιστής να αρχίσει να χρησιμοποιεί την *virtual memory* που στην ουσία είναι ο διαθέσιμος χώρος του σκληρού δίσκού για να μπορέσει να φέρει εις πέρας την διαδικασία. Αυτό θα οδηγήσει σε χρονικά αργότερη περάτωση της διαδικασίας ή στο χειρότερο σενάριο μπορεί το πρόγραμμα μας να μην εκτελεστεί.

Παρατηρήσαμε επίσης πως η νόρμα L_2 για για τα ορθογώνια πλέγματα των παραδειγμάτων 5,6 και 7 μειώνεται μονοτονικά όσο αυξάνεται ο πολυωνυμικός βαθμός p. Όπως ήταν λογικό λόγω της αύξησης του p και της ταυτόχρονης πύκνωσης του πλέγματος ο χρόνος εκτέλεσης των αλγορίθμων τείνει να αυξηθεί εκθετικά. Όπως αναφέραμε και στο τέλος του προηγούμενου κεφαλαίου για το ορθογώνιο πλέγμα του Παραδείγματος 8 που είδαμε, η νόρμα L_2 δεν διαφέρει ουσιαστικά ουσιαστικά από αυτή του Παραδείγματος 7, χρειάστηκε όμως μετατροπή και επέκταση στον αρχικό αλγόριθμο για να μας δώσει τα τοπικα κομβοδιανύσματα καθώς προγραμματιστικά ήταν απαραίτητη μία ισχυρότερη λογική για την εύρεση αυτών.

7.1 Μελλοντική δουλειά

Η παρούσα μεταπτυχιακή εργασία υλοποίησε την μέθοδο της Ισογεωμετρικής Ανάλυσης στο προγραμματιστικό περιβάλλον Matlab. Για ακαδημαϊκούς σκοπούς αυτό δεν αποτελεί κανένα απολύτως πρόβλημα γιατί πρωταρχικός μας στόχος ήταν τα προγράμ-

ματα που φτιάξαμε να δουλεύουν επιτυχώς και η χρησιμοποίηση έτοιμων υπορουτινών που διαθέτει το Matlab. Σε πραγματικές συνθήκες εργασίας όμως δεν θα ίσχυε το ίδιο γι΄ αυτό θα έπρεπε να φροντίσουμε οι αλγόριθμοί μας να είναι γραμμένοι με το βέλτιστο δυνατό τρόπο και επίσης ίσως να χρησιμοποιηθεί μια ταχύτερη υπολογιστικά γλώσσα προγραμματισμού όπως π.χ η C++ ή και άλλες.

Ως παράδειγμα αξίζει να αναφέρουμε πως ένα απλός πολλαπλασιασμός στοιχείων πινάκων στο Matlab όπως ο $A(i) = B(i) * C(i)$ για $i = 1$ έως n μπορεί να είναι από 10 έως και 100 φορές αργότερος από τον ίδιο κώδικα στην C++. Σε πολύπλοκα και μεγάλα προγράμματα τέτοιες πράξεις είναι δύσκολο να αποφευχθούν.

Τέλος όλα τα παραπάνω βασίζονται και υλοποιούνται προγαμματιστικά πάνω σε προβλήματα δύο διαστάσεων καθώς οι έννοιες είναι πιο ευκολες να κατανοηθούν στο διδιάστατο χώρο. Βέβαια για όποιον το επιθυμεί όλα αυτά μπορούν να επεκταθούν και στον χώρο των τριών διαστάσεων αλλά για αυτή την διπλωματική όπως εξηγήσαμε δεν ήταν απαραίτητο.

Appendix A

Codes

Here, an algorithm for T-splines is presented that starts by having as input a set of knots and patches. Six examples of T-meshes are used that the user can freely choose from, along with the polynomial degree.

Given the initial data, the output of the program gives us the local knot vectors in each direction, the basis functions, the partial derivatives and their plots. This program is constructed for 2D T-meshes and can be generalized for 3D cases.

```
1  clear all;
2  clc;
3
4  %% John Raditsas
5  %---------------------------------- Start
6  example = input('Choose <1,2,3,4,5,6> of example to be solved
       : '); % Choose example
7  p = input('Choose polynomial degree: '); % Choose p
8  P = calculateP(p);  % Vector length
9
10 if(example == 1)
11
12     %---------------------------------- Sample 1
13     % Mesh points
14     points = [
15         0 0
16         1 0
17         2 0
18         0 1
19         1 1
20         0 2
21         1 2
22         2 2
23     ];
24
25     % Below are the boundary points of each patch.
26     % Each patch consists of 4 points.
27     patches = [
28         0 0 1 0 1 1 0 1
```

```
        1 0 2 0 2 2 1 2
        0 1 1 1 1 2 0 2
    ];

elseif (example == 2)

    %---------------------------------- Sample 2
    % Mesh points
    points = [
        0 0
        1 0
        3 0
        1 1
        2 1
        3 1
        0 2
        1 2
        0 3
        1 3
        2 3
        3 3
    ];

    % Below are the boundary points of each patch.
    % Each patch consists of 4 points.
    patches = [
        0 0 1 0 1 2 0 2
        1 0 3 0 3 1 1 1
        2 1 3 1 3 3 2 3
        1 1 2 1 2 3 1 3
        0 2 1 2 1 3 0 3
    ];

elseif (example == 3)

    %---------------------------------- Sample 3
    % Mesh points
    points = [
        0 0
        1 0
        2 0
        3 0
        5 0
        0 1
        1 1
        2 1
        3 1
        4 1
```

```
        5 1
        0 2
        1 2
        3 2
        0 3
        1 3
        3 3
        4 3
        5 3
        0 4
        3 4
        4 4
        5 4
    ];

    % Below are the boundary points of each patch.
    % Each patch consists of 4 points.
    patches = [
        0 0 1 0 1 1 0 1
        1 0 2 0 2 1 1 1
        2 0 3 0 3 1 2 1
        3 0 5 0 5 1 3 1
        0 1 1 1 1 2 0 2
        1 1 3 1 3 2 1 2
        3 1 4 1 4 4 3 4
        4 1 5 1 5 3 4 3
        0 2 1 2 1 3 0 3
        1 2 3 2 3 3 1 3
        0 3 3 3 3 4 0 4
        4 3 5 3 5 4 4 4
    ];

elseif (example == 4)

    %----------------------------------- Sample 4
    % Mesh points
    points = [
        1 1
        3 1
        4 1
        6 1
        7 1
        1 3
        3 3
        4 3
        1 4
        2 4
        3 4
```

```
125             4  4
126             5  4
127             6  4
128             7  4
129             1  5
130             2  5
131             3  5
132             4  5
133             5  5
134             1  6
135             3  6
136             5  6
137             1  7
138             5  7
139             7  7
140         ];
141
142         % Below are the boundary points of each patch.
143         % Each patch consists of 4 points.
144         patches = [
145             1  1  3  1  3  3  1  3
146             3  1  4  1  4  3  3  3
147             4  1  6  1  6  4  4  4
148             6  1  7  1  7  4  6  4
149             1  3  3  3  3  4  1  4
150             3  3  4  3  4  4  3  4
151             1  4  2  4  2  5  1  5
152             2  4  3  4  3  5  2  5
153             3  4  4  4  4  5  3  5
154             4  4  5  4  5  5  4  5
155             5  4  7  4  7  7  5  7
156             1  5  3  5  3  6  1  6
157             3  5  5  5  5  6  3  6
158             1  6  5  6  5  7  1  7
159         ];
160
161     elseif(example == 5)
162
163         %----------------------------------- Sample 5
164         % Mesh points
165         points = [
166             0  0
167             0  0
168             0  0
169             1  0
170             1  0
171             1  0
172             0  0
```

```
        0 0
        0 0
        1 0
        1 0
        1 0
        0 0
        0 0
        0 0
        1 0
        1 0
        1 0
        0 1
        0 1
        0 1
        1 1
        1 1
        1 1
        0 1
        0 1
        0 1
        1 1
        1 1
        1 1
        0 1
        0 1
        0 1
        1 1
        1 1
        1 1
    ];

    % Below are the boundary points of each patch.
    % Each patch consists of 4 points.
    patches = [
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 1 0 1 0 0 0
        1 0 1 0 1 0 1 0
        1 0 1 0 1 0 1 0
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 1 0 1 0 0 0
        1 0 1 0 1 0 1 0
        1 0 1 0 1 0 1 0
        0 0 0 0 0 1 0 1
        0 0 0 0 0 1 0 1
        0 0 1 0 1 1 0 1
        1 0 1 0 1 1 1 1
```

```
          1  0  1  0  1  1  1  1
          0  1  0  1  0  1  0  1
          0  1  0  1  0  1  0  1
          0  1  1  1  1  1  0  1
          1  1  1  1  1  1  1  1
          1  1  1  1  1  1  1  1
          0  1  0  1  0  1  0  1
          0  1  0  1  0  1  0  1
          0  1  1  1  1  1  0  1
          1  1  1  1  1  1  1  1
          1  1  1  1  1  1  1  1
      ];

elseif(example == 6)

      %----------------------------------- Sample 6
      % Mesh points
      points = [
          0  0
          0  0
          0  0
        0.5  0
          1  0
          1  0
          1  0
          0  0
          0  0
          0  0
        0.5  0
          1  0
          1  0
          1  0
          0  0
          0  0
          0  0
        0.5  0
          1  0
          1  0
          1  0
          0  0.5
          0  0.5
          0  0.5
        0.5  0.5
          1  0.5
          1  0.5
          1  0.5
          0  1
          0  1
```

```
        0 1
      0.5 1
        1 1
        1 1
        1 1
        0 1
        0 1
        0 1
      0.5 1
        1 1
        1 1
        1 1
        0 1
        0 1
        0 1
      0.5 1
        1 1
        1 1
        1 1
    ];

    % Below are the boundary points of each patch.
    % Each patch consists of 4 points.
    patches = [
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 0.5 0 0.5 0 0 0
        0.5 0 1 0 1 0 0.5 0
        1 0 1 0 1 0 1 0
        1 0 1 0 1 0 1 0
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 0.5 0 0.5 0 0 0
        0.5 0 1 0 1 0 0.5 0
        1 0 1 0 1 0 1 0
        1 0 1 0 1 0 1 0
        0 0 0 0 0 0.5 0 0.5
        0 0 0 0 0 0.5 0 0.5
        0 0 0.5 0 0.5 0.5 0 0.5
        0.5 0 1 0 1 0.5 0.5 0.5
        1 0 1 0 1 0.5 1 0.5
        1 0 1 0 1 0.5 1 0.5
        0 0.5 0 0.5 0 1 0 1
        0 0.5 0 0.5 0 1 0 1
        0 0.5 0.5 0.5 0.5 1 0 1
        0.5 0.5 1 0.5 1 1 0.5 1
        1 0.5 1 0.5 1 1 1 1
        1 0.5 1 0.5 1 1 1 1
```

```
317         0 1 0 1 0 1 0 1
318         0 1 0 1 0 1 0 1
319         0 1 0.5 1 0.5 1 0 1
320         0.5 1 1 1 1 1 0.5 1
321         1 1 1 1 1 1 1 1
322         1 1 1 1 1 1 1 1
323         0 1 0 1 0 1 0 1
324         0 1 0 1 0 1 0 1
325         0 1 0.5 1 0.5 1 0 1
326         0.5 1 1 1 1 1 0.5 1
327         1 1 1 1 1 1 1 1
328         1 1 1 1 1 1 1 1
329     ];
330
331 else
332     return;
333 end
334
335 Us = []; % Total U elements from points
336 Vs = []; % Total V elements from points
337
338 for i=1:length(points)
339     Us = [Us, points(i,1)];
340     Vs = [Vs, points(i,2)];
341 end
342
343 Us = unique(Us); % Remove duplicate U elements
344 Vs = unique(Vs); % Remove duplicate V elements
345
346 % Boundaries
347 Umax = max(Us); % Get the max U from all points
348 Umin = min(Us); % Get the min U from all points
349 Vmax = max(Vs); % Get the max V from all points
350 Vmin = min(Vs); % Get the min U from all points
351
352 % Final vectors initialization
353 U_vec = [];
354 V_vec = [];
355
356 if (detectAnomaly(patches) == true)
357     % exceptional case
358
359     [U_vec, V_vec] = getAnomalyPoints(points, patches, P);
360
361 else
362     % other cases
363
364     if (mod(P,2) ~= 0)
```

```
365
366         % When p is odd
367         for x=1:length(points)
368             U_vec = [U_vec; getPointsOnUAxis(points, points(x
                ,:), P, Umin, Umax)];
369             V_vec = [V_vec; getPointsOnVAxis(points, points(x
                ,:), P, Vmin, Vmax)];
370         end
371
372     else
373
374         % When p is even
375         for y=1:size(patches,1)
376             center = findRectangleCenter(patches(y,:));
377             U_vec = [U_vec; getPointsOnUAxis(points, center,
                P, Umin, Umax, patches(y,:))];
378             V_vec = [V_vec; getPointsOnVAxis(points, center,
                P, Vmin, Vmax, patches(y,:))];
379         end
380
381     end
382
383 end
384
385 % Printing final vectors
386
387 U_vec
388 V_vec
389
390 %% Check the calculated local knot arrays (U_vec, V_vec):
391     if(size(U_vec,1) ~= size(V_vec,1))
392         fprintf('*** PROBLEM FOUND IN U_vec & V_vec ***\n');
393         return
394     end
395 %% Calculate and plot the basis functions:
396     xmin=min(reshape(points(:,1),size(points,1),1));
397     xmax=max(reshape(points(:,1),size(points,1),1));
398     ymin=min(reshape(points(:,2),size(points,1),1));
399     ymax=max(reshape(points(:,2),size(points,1),1));
400     numberOfIntersections = size(points,1);
401     numberOfPatches = size(patches,1);
402 % Generalization:
403     numberOfAnchors = size(U_vec,1);    %Under question
404 %%
405 figure(1)
406 % plot and number knot intersections:
407 ds=(xmax-xmin)/50;
408     for i=1:numberOfIntersections
```

```
409         hold on
410         xintsect=points(i,1);
411         yintsect=points(i,2);
412         plot(xintsect,yintsect,'bo')
413         text(xintsect+ds,yintsect+ds,int2str(i),'FontSize'
               ,10,'Color','red','FontWeight','Bold')
414     end
415
416 % plot patches (and anchor numbers at element centroids, if p
      =2,4,6,...):
417 for i=1:numberOfPatches
418     %Corner points:
419     xA=patches(i,1); yA=patches(i,2);
420     xB=patches(i,3); yB=patches(i,4);
421     xC=patches(i,5); yC=patches(i,6);
422     xD=patches(i,7); yD=patches(i,8);
423     %Centroid:
424     xc=0.25*(xA+xB+xC+xD);
425     yc=0.25*(yA+yB+yC+yD);
426     hold on
427     xloc=[xA xB xC xD xA]; yloc=[yA yB yC yD yA];
428     plot(xloc,yloc,'b')
429     hold on
430     %Centroid is shown only when it is an anchor (p=even),
          otherwise only numbered:
431     if(mod(p,2) == 0)
432         plot(xc,yc,'ro', 'MarkerFaceColor', 'r')
433     end
434     text(xc+ds,yc+ds,int2str(i),'FontAngle','italic','
          FontSize',11,'FontWeight','Bold')
435 end
436 axis equal
437 %--------------
438 hold off
439 %--------------
440
441 %% Based on the above "U_vec" and "V_vec", construct the
      local
442 % B-slines and then the normalization:
443
444 %% Determine the step we move horizontally and vertically
      from anchor:
445     nc=0;
446     if(mod(p,2)==0)
447         step=p/2+1;
448         ncolumns=2*step;
449     else
450         step=(p+1)/2;
```

```
451         ncolumns=2*step+1;
452     end
453 %---Comply the commands below (older) with the above (new)
       symbols:
454     k = p + 1;           %polynomial order
455 %---Divisions of the parametric space:
456     NsubdivisionsX = 30;     %the user may change it!
457     NsubdivisionsY = 30;     %the user may change it!
458     taux = linspace(xmin,xmax,NsubdivisionsX+1);
459     tauy = linspace(ymin,ymax,NsubdivisionsY+1);
460 %-------------
461     nanchors = numberOfAnchors;
462 %-------------
463 %%LOOP ON ALL THE ANCHORS
464     for ianchor=1:nanchors
465             knotsLocy=V_vec(ianchor,1:ncolumns); %(was 4 for
                   p=2)
466                 nstep=2; %we need only two terms(Nip and
                       first derivative).
467                 colmaty=spcol(knotsLocy,p+1,brk2knt(tauy,
                       nstep)); %All basis functions Nip and (p
                       -1) derivatives.
468                 [i1y,i2y]=size(colmaty);
469                 Basisy = colmaty(1:nstep:i1y,:);
470                 dBasisy= colmaty(2:nstep:i1y,:);   %first
                       derivative over y
471             knotsLocx=U_vec(ianchor,1:ncolumns); %(was 4 for
                   p=2)
472                 colmatx=spcol(knotsLocx,p+1,brk2knt(taux,
                       nstep)); %All basis functions Nip and (p
                       -1) derivatives.
473                 [i1x,i2x]=size(colmatx);
474                 Basisx = colmatx(1:nstep:i1x,:);
475                 dBasisx= colmatx(2:nstep:i1x,:);   %first
                       derivative over x
476             for ix=1:length(tauy)
477                 for jy=1:length(taux)
478                     Nip(ianchor,ix,jy)=Basisx(jy)*Basisy(ix);
479                     dNip_dx(ianchor,ix,jy)=dBasisx(jy)*Basisy
                           (ix);   %dN/dx
480                     dNip_dy(ianchor,ix,jy)=Basisx(jy)*dBasisy
                           (ix);   %dN/dy
481                 end
482             end
483 %             plot(taux,Basisx)
484     end
485 %
486 %% Normalize the basis functions
```

```
487     weights(1:nanchors) = 1;  %%%XYWctrl(1:nanchors,3);
488     for i=1:nanchors
489         for ix=1:length(tauy)
490             for jy=1:length(taux)
491                 denom=0; denom_X=0; denom_Y=0;
492                 for j=1:nanchors
493                     denom = denom + Nip(j,ix,jy)*weights(j);
494                     denom_X = denom_X + dNip_dx(j,ix,jy)*
                           weights(j);
495                     denom_Y = denom_Y + dNip_dy(j,ix,jy)*
                           weights(j);
496                 end
497 %                   fprintf('Nip=%10.4f   denom=%10.4f\n',Nip(
    i,ix,jy),denom);
498                     if(Nip(i,ix,jy) == 0)
499                         Rip(i,ix,jy)=0;
500                         derRip_X(i,ix,jy)=(dNip_dx(i,ix,jy)*
                               weights(i)-...
501                                                 Nip(i,ix,jy)*
                                                       denom_X)/
                                                       denom^2;
502                         derRip_Y(i,ix,jy)=(dNip_dy(i,ix,jy)*
                               weights(i)-...
503                                                 Nip(i,ix,jy)*
                                                       denom_Y)/
                                                       denom^2;
504                     else
505                         Rip(i,ix,jy)=Nip(i,ix,jy)*weights(i)/
                               denom;
506                         derRip_X(i,ix,jy)=(dNip_dx(i,ix,jy)*
                               weights(i)-...
507                                                 Nip(i,ix,jy)*
                                                       denom_X)/
                                                       denom^2;
508                         derRip_Y(i,ix,jy)=(dNip_dy(i,ix,jy)*
                               weights(i)-...
509                                                 Nip(i,ix,jy)*
                                                       denom_Y)/
                                                       denom^2;
510                     end
511             end
512         end
513     end
514 % %Check partition of unity:
515  choice = input('*** Do you want to check Partition of Unity
       ?: YES<1> NO<0> ');
516  if(choice == 1)
517      npoint=0;
```

```
518       for ix=1:length(tauy)
519           for jy=1:length(taux)
520               npoint=npoint+1;
521               sumRip = 0;
522               for i=1:nanchors
523                   sumRip = sumRip + Rip(i,ix,jy);
524               end
525               fprintf('Point %3i   sumRip=%10.4f\n',npoint,
                      sumRip);
526           end
527       end
528   end
529   %
      ********************************************************************

530   %% Plot the normalized T-splines:
531   dx=(xmax-xmin)/(length(taux)-1);
532   dy=(ymax-ymin)/(length(tauy)-1);
533   for i=1:length(tauy)
534       for j=1:length(taux)
535           X(i,j)=xmin+(j-1)*dx;
536           Y(i,j)=ymin+(i-1)*dy;
537       end
538   end
539   %
      --------------------------------------------------------------------

540   figure(2)
541   RipPlot=zeros(length(tauy),length(taux));
542   for ip=1:nanchors
543      RipPlot(1:length(tauy),1:length(taux))= Rip(ip,1:length(
          tauy),1:length(taux));
544      surface(X,Y,RipPlot)
545      hold on
546   end
547   xlabel('X')
548   ylabel('Y')
549   zlabel('Rip')
550   title('BASIS FUNCTIONS')
551   %
      --------------------------------------------------------------------

552   figure(3)
553   RipPlot=zeros(length(tauy),length(taux));
554   for ip=1:nanchors
555      RipPlot(1:length(tauy),1:length(taux))= derRip_X(ip,1:
          length(tauy),1:length(taux));
556      surface(X,Y,RipPlot)
```

```
557     hold on
558 end
559 xlabel('X')
560 ylabel('Y')
561 zlabel('Rip')
562 title('X-PARTIAL DERIVATIVE OF BASIS FUNCTIONS')
563 %
    **********************************************************************
```

Below, we present the main modified part of the programm that produces the local knot vectors for the T-mesh of example 8.

```
% The middle cell of our T-mesh for example 8
%
%
%       o-----o
%       |     |
%       |-----|
%       |     |
%       o-----o
%
%

[points, patches, nan_points] = getData( p ); % Data
   initialization

Us = []; % Total U elements from points
Vs = []; % Total V elements from points

for i=1:length(points)
    Us = [Us, points(i,1)];
    Vs = [Vs, points(i,2)];
end

Us = unique(Us); % Remove duplicate U elements
Vs = unique(Vs); % Remove duplicate V elements

step_U = min(diff(sort(Us))); % Calculating minimum U step
   iteration depending on uniq Us
step_V = min(diff(sort(Vs))); % Calculating minimum V step
   iteration depending on uniq Vs

% Boundaries
Umax = max(Us); % Get the max U from all points
Umin = min(Us); % Get the min U from all points
Vmax = max(Vs); % Get the max V from all points
Vmin = min(Vs); % Get the min U from all points

% Final vectors initialization
U_vec = [];
V_vec = [];

if (example == 8) % Or similar T-meshes with example 8

    if (mod(P,2) == 0)

        [U_res, V_res] = calculateEvenMesh(patches, p);

```

```
44          U_vec = U_res;
45          V_vec = V_res;
46
47      else
48
49          [U_res, V_res] = calculateOddMesh(nan_points, p);
50
51          U_vec = U_res;
52          V_vec = V_res;
53
54      end
55
56  elseif ((detectAnomaly(patches) == true))
57
58      % exceptional case
59      [U_vec, V_vec] = getAnomalyPoints(points, patches, P,
            true, Us, Vs);
60
61  else
62      % other cases
63
64      if (mod(P,2) ~= 0)
65
66          % When p is odd
67          for x=1:length(points)
68              U_vec = [U_vec; getPointsOnUAxis(step_U, points,
                    points(x,:), P, Umin, Umax)];
69              V_vec = [V_vec; getPointsOnVAxis(step_V, points,
                    points(x,:), P, Vmin, Vmax)];
70          end
71
72      else
73
74          % When p is even
75          for y=1:size(patches,1)
76              center = findRectangleCenter(patches(y,:));
77              U_vec = [U_vec; getPointsOnUAxis(step_U, points,
                    center, P, Umin, Umax, patches(y,:))];
78              V_vec = [V_vec; getPointsOnVAxis(step_V, points,
                    center, P, Vmin, Vmax, patches(y,:))];
79          end
80
81      end
82
83  end
```

Here, we are pre-calculating the Bezier-Bernstein polynomials in the x axis for $p = 3$ used for the Bézier extraction method. This algorithm can be further generalized for higher polynomial degrees by manually expanding it or using the univariate extraction operator function below.

```
clear all;
clc;

%% PreCalculating the Bezier-Bernstein polynomials in x
   direction for p=3

% Initial values
a = 3;    % length of x axis
p = 3;    % polynomial degree
nx = 3;   % number of elements in x direction

% Gauss Points and weights
[gi,~] = pgau_rectangle_FULL;

% Values at GPTs:
ngaussx = p+1;
ngaussy = p+1;

% X-direction:
nc=0;
dxgau=a/nx;
for icel=1:nx
    for jg=1:ngaussx
        nc=nc+1;
        xmid=dxgau/2+(icel-1)*dxgau;
        tauGAUx(nc)=xmid+gi(jg,ngaussx)*(dxgau/2);
    end
end

nc=0;
bezier(1:12,1:6)=0;
for icel=1:3
    for ig=1:ngaussx
        nc=nc+1;
        if(icel==1)
            t=tauGAUx(nc);
            B1=(1-t)^3; B2=3*(1-t)^2*t; B3=3*(1-t)*t^2; B4=t
               ^3;
            dB1=-3*(1-t)^2; dB2=3*(1-4*t+3*t^2); dB3=3*(2*t
               -3*t^2); dB4=3*t^2;
            bezier(nc,1)=B1;                    dbezier(nc,1)
               =dB1;
            bezier(nc,2)=1*B2+1/2*B3+1/4*B4;    dbezier(nc,2)
               =1*dB2+1/2*dB3+1/4*dB4;
```

```
40              bezier(nc,3)=     1/2*B3+7/12*B4;   dbezier(nc,3)
                   =     1/2*dB3+7/12*dB4;
41              bezier(nc,4)=              1/6*B4;   dbezier(nc,4)
                   =              1/6*dB4;
42          elseif(icel==2)
43              t=tauGAUx(nc)-1;
44              B4=(1-t)^3; B5=3*(1-t)^2*t; B6=3*(1-t)*t^2; B7=t
                   ^3;
45              dB4=-3*(1-t)^2; dB5=3*(1-4*t+3*t^2); dB6=3*(2*t
                   -3*t^2); dB7=3*t^2;
46              bezier(nc,2)=1/4*B4;
                   dbezier(nc,2)=1/4*dB4;
47              bezier(nc,3)=7/12*B4+2/3*B5+1/3*B6+1/6*B7;
                   dbezier(nc,3)=7/12*dB4+2/3*dB5+1/3*dB6+1/6*dB7
                   ;
48              bezier(nc,4)=1/6 *B4+1/3*B5+2/3*B6+7/12*B7;
                   dbezier(nc,4)=1/6 *dB4+1/3*dB5+2/3*dB6+7/12*
                   dB7;
49              bezier(nc,5)=                         1/4*B7;
                   dbezier(nc,5)=                              1/4*dB7
                   ;
50          else
51              t=tauGAUx(nc)-2;
52              B7=(1-t)^3; B8=3*(1-t)^2*t; B9=3*(1-t)*t^2; B10=t
                   ^3;
53              dB7=-3*(1-t)^2; dB8=3*(1-4*t+3*t^2); dB9=3*(2*t
                   -3*t^2); dB10=3*t^2;
54              bezier(nc,3)=1/6*B7;                          dbezier
                   (nc,3)=1/6*dB7;
55              bezier(nc,4)=7/12*B7+1/2*B8+0*B9+0*B10;   dbezier
                   (nc,4)=7/12*dB7+1/2*dB8+0*dB9+0*dB10;
56              bezier(nc,5)=1/4*B7+1/2*B8+1*B9+0*B10;    dbezier
                   (nc,5)=1/4*dB7+1/2*dB8+1*dB9+0*dB10;
57              bezier(nc,6)=1*B10;                           dbezier
                   (nc,6)=1*dB10;
58          end
59      end
60  end
```

The function below computes the univarite extraction operator C. Similar functions of the extraction operator can be found in other papers or in the IGAFEM library.

```
% Output: C(:,:,e) - extraction operators, e=1,2,..,nel+1 in xi direction
% C(:,:,nel+1) - Identity matrix due to the algorithm, not used
%
% Input: knotxi - knot vector
% p - polynomial order

function C=ExtractionOperator(knotxi,p)

% Initializations
m=length(knotxi);
a=p+1;
b=a+1;
nb=1;
C(:,:,1)=eye(p+1);

    while b<m
    C(:,:,nb+1)=eye(p+1); % Initialize the next extraction operator
    i=b;

    % Count multiplicity of the knot at location b
    while b<m && knotxi(b+1)==knotxi(b)
        b=b+1;
    end
        mult=b-i+1;
    if mult<p
        numer=knotxi(b)-knotxi(a);
        for j=p:-1:mult+1
            alphas(j-mult)=numer/(knotxi(a+j)-knotxi(a));
        end
        r=p-mult;

    % Update the matrix coefficient for r new knots
        for j=1:r
            save=r-j+1;
            s=mult+j;

            for k=p+1:-1:s+1
                alpha=alphas(k-s);
                % Form extraction operator
                C(:,k,nb)=alpha*C(:,k,nb)+(1-alpha)*C(:,k-1,nb);
            end

```

```
43              if b<m
44                  % Update overlapping coefficients of the next
                        operator
45                  C(save:j+save,save,nb+1)=C(p-j+1:p+1,p+1,nb);
46              end
47          end
48
49          % Finished with the current operator.
50          % Update indices for the next operator.
51           nb=nb+1;
52           if b<m
53              a=b;
54              b=b+1;
55           end
56
57      elseif mult==p % In case multiplicity of knot is already
           p,
58
59          % update indices for the next operator.
60          nb=nb+1;
61          if b<m
62              a=b;
63              b=b+1;
64          end
65      end
66  end
67
68  end
```

Bibliography

[1] Michael J. Borden, Michael A. Scott, John A. Evans, and Thomas J. R. Hughes. Isogeometric Finite Element Data Structures based on Bezier Extraction of NURBS. *International Journal for Numerical Methods in Engineering*, 87(1-5):15-47, 2011.

[2] Michael A. Scott, Michael J. Borden, Clemens V. Verhoosel, Thomas W. Sederberg, and Thomas J. R. Hughes. Isogeometric finite element data structures based on Bezier extraction of T-splines. *International Journal for Numerical Methods in Engineering*, 88(2):126-156, 2011.

[3] W. Boehm. Inserting new knots into B-spline curves. *Computer-Aided Design*, 12(4):199-201, 1980.

[4] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM Transactions on Graphics (TOG)*, 23(3):276-283, 2004.

[5] Li, X. & Scott, Michael. Analysis-suitable T-splines: Characterization, refineability, and approximation. *Mathematical Models and Methods in Applied Sciences*, 24. 10.1142/S0218202513500796, 2012.

[6] J. Austin Cottrell, Thomas J.R. Hughes, Yuri Bazilevs. *Isogeometric Analysis Towards Integration of CAD and FEA*. John Wiley and Sons, Ltd, West Sussex, 2009.

[7] Dr. Les Piegl, Dr. Wayne Tiller (auth.). *The NURBS Book: Monographs in Visual Communications*, 2nd edition, Springer-Verlag Berlin Heidelberg, 1997.

[8] Gernot Beer, Stéphane Bordas (eds.). *Isogeometric Methods for Numerical Simulation*, CISM International Centre for Mechanical Sciences 561, Springer-Verlag Wien, 2015.

[9] R.W. Clough. The finite element method in plane stress analysis. Proceedings of 2nd ASCE Conference on Electronic Computation, Pittsburgh Pa., Sept. 8 and 9, 1960.

[10] TJR Hughes, JA Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39-41):4135-4195, 2005.

[11] T.W. Sederberg & J. Zheng & A. Bakenov & A. Nasri. T-splines and TNURCCs. *ACM Transactions on graphics* , 22(3), pp. 477-484, 2003.

[12] Narayan, K. Lalit *Computer Aided Design and Manufacturing*. New Delhi: Prentice Hall of India. ISBN 978-8120333420, 2008.

[13] Duggal, Vijay. *Cadd Primer: A General Guide to Computer Aided Design and Drafting-Cadd, CAD*. Mailmax Pub. ISBN 978-0962916595, 2000.

[14] M.A. Scott, X. Li, T.W. Sederberg & T.J.R. Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213:206-222, 2012.

[15] Runge, Carl, *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*, Zeitschrift für Mathematik und Physik, 46: 224–243, 1901.

[16] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, et al.. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, 199, pp. 229-263, 2010.

[17] J. H. Argyris. Energy Theorems and Structural Analysis: A Generalized Discourse with Applications on Energy Principles of Structural Analysis Including the Effects of Temperature and Non-Linear Stress-Strain Relations. *AIRCRAFT ENGINEERING* , 26(10): 347-356, 1955.

[18] C.G. Provatidis. *Precursors of Isogeometric Analysis*, Springer, Cham, 2019.

[19] Χ. Προβατίδης. *Πεπερασμένα Στοιχεία στην Ανάλυση Κατασκευών*, 2η έκδοση, Εκδόσεις Τζιόλα, Θεσσαλονίκη, 2017.