ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Αποσύνθεση Bézier για παρεμβολή T-spline σε διδιάστατα προβλήματα πεδίου

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

Ζησιμοπούλου Πολυτίμης

Επιβλέπων: Προβατίδης Χριστόφορος
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2022

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας μεταπτυχιακής εργασίας είναι η παρουσίαση του θεωρητικού υποβάθρου της Ισογεωμετρικής Ανάλυσης και των υπολογιστικών μεθόδων της, στην επίλυση βαθμωτών προβλημάτων συνοριακών τιμών (ΠΣΤ) καθώς και η υλοποίηση των εν λόγω μεθόδων στο προγραμματιστικό περιβάλλον Matlab. Η θεωρητική τεκμηρίωση περιλαμβάνει την παρουσίαση των θεμελιακών γεωμετρικών συναρτήσεων NURBS και B-spline, αλλά και των διαδόχων τους, των συναρτήσεων T-spline. Επιπλέον, αναλύεται η καταλληλότητα των T-spline να ορίσουν βάση για το χώρο λύσεων του ΠΣΤ. Παρουσιάζεται, επίσης, η Αποσύνθεση Bézier των παραπάνω συναρτήσεων στη μονοδιάστατη και στη διδιάστατη περίπτωση και η εφαρμογή της στην επίλυση διαφορικών εξισώσεων, ως εναλλακτική μέθοδος έναντι της συμβατικής Ισογεωμετρικής Ανάλυσης.

Η υπολογιστική εφαρμογή των παραπάνω μεθοδολογιών αφορά την κατασκευή δυο γεωμετρικών μοντέλων-ενός ορθογώνιου χωρίου και ενός καμπυλόγραμμου- με τη χρήση πλεγμάτων NURBS και T-spline και την μετέπειτα χρήση των εν λόγω γεωμετρικών συναρτήσεων για την επίλυση του χρονο-ανεξάρτητου προβλήματος μεταφοράς θερμότητας στα προαναφερθέντα χωρία, με μεικτές συνοριακές συνθήκες (μη ομογενείς Dirichlet και Neumann). Επιπλέον, και για τις δυο κατηγορίες πλεγμάτων, κατασκευάστηκαν αλγόριθμοι για την υλοποίηση της Αποσύνθεσης Bézier και τον υπολογισμό των τοπικών τελεστών εξαγωγής, με βάση τους οποίους υπολογίστηκαν στη συνέχεια οι συναρτήσεις NURBS/T-spline, σύμφωνα με την κλασική διαδικασία εφαρμογής της Αποσύνθεσης Bézier στην Ισογεωμετρική Ανάλυση. Για την πλήρη διερεύνηση της Αποσύνθεσης Bézier, το πρόβλημα συνοριακών τιμών επιλύθηκε εκ νέου, στην περίπτωση του καμπύλου χωρίου NURBS, με τις συναρτήσεις Bernstein του εκλεπτυσμένου, πλέον, πλέγματος. Κάθε μια από τις παραπάνω μεθοδολογίες εξετάστηκε ως προς την ακρίβεια της αριθμητικής λύσης της, με βάση την νόρμα L_2 του σφάλματος της προσέγγισης. Τέλος, παρατίθενται τα συμπεράσματα της έρευνας, όπως εξήχθησαν μέσα από τις εν λόγω υπολογιστικές εφαρμογές και προτείνονται βελτιώσεις, με αφορμή τις παρατηρήσεις που προέκυψαν.

ABSTRACT

The aim of this thesis is the presentation of the theoretical background of Isogeometric Analysis and its computational methods in the solution of scalar boundary value problems (BVP) as well as the implementation of these methods in the Matlab environment. The theoretical documentation includes the presentation of the fundamental geometric functions NURBS and B-splines, and their successors, the T-spline functions. In addition, the suitability of T-splines to define a basis for the solution space of the BVP is analyzed. It also presents the Bézier Decomposition of the above functions in the one-dimensional and two-dimensional case and its application to the solution of differential equations, as an alternative method to the conventional Isogeometric Analysis.

The computational application of the above methods involves the construction of two geometric models - a rectangular domain and a curved domain - using NURBS and T-spline meshes and the subsequent use of these geometric functions to solve the time-independent heat transfer problem in the aforementioned domains, with mixed boundary conditions (non-homogeneous Dirichlet and Neumann). In addition, for both categories of meshes, algorithms were constructed for the implementation of the Bézier Decomposition and the calculation of the local extraction operators, on the basis of which the NURBS/T-spline functions were then calculated, according to the classical procedure for the application of the Bézier Decomposition in isogeometric analysis. To fully investigate the Bézier Decomposition, the boundary value problem was re-solved, in the case of the curved NURBS space, with the Bernstein functions of the now refined mesh. Each of the above methodologies was tested for the accuracy of its numerical solution, based on the L_2 norm of the approximation error. Finally, the conclusions of the research, as extracted through these computational applications, are presented and improvements are proposed based on the observations made.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή, κ. Χριστόφορο Προβατίδη, με την καθοδήγηση του οποίου ανακάλυψα με ενθουσιασμό το αντικείμενο της Ισογεωμετρικής Ανάλυσης καθώς και για την αγαστή συνεργασία που είχαμε καθ' όλη τη διάρκεια εκπόνησης της μεταπτυχιακής μου εργασίας. Επίσης, θα ήθελα να εκφράσω την αμέριστη ευγνωμοσύνη μου στους γονείς μου και τον αδερφό μου για την ανιδιοτελή αρωγή τους στην εκπλήρωση των ακαδημαϊκών μου στόχων, καθώς και τους φίλους μου, Ελευθερία και Νίκο, για την στήριξη τους όλο το διάστημα της μελέτης μου.

ΠΕΡΙΕΧΌΜΕΝΑ

ΛΊΣΤΑ ΣΧΗΜΆΤΩΝ

1 ΕΙΣΑΓΩΓΗ

Η Μέθοδος Πεπερασμένων Στοιχείων (Finite Element Method, FEM) ή αλλιώς η Ανάλυση Πεπερασμένων Στοιχείων (Finite Element Analysis, FEA) θεωρείται η κατ' εξοχήν μέθοδος για την αριθμητική επίλυση διαφορικών εξισώσεων, οι οποίες περιγράφουν προβλήματα των φυσικών επιστημών και της μηχανικής. Η περιοχή του χώρου, στην οποία λαμβάνουν χώρα τα προβλήματα αυτά, διαιρείται σε ένα πεπερασμένο πλήθος στοιχείων, πάνω στα οποία προσεγγίζεται αριθμητικά η λύση. Αφ'ενός, η μοντελοποίηση του χωρίου εφαρμογής του προβλήματος συνοριακών τιμών καλείται *Σχεδίαση με τη βοήθεια Η/Υ* (Computer-aided design, CAD) και αποτελεί μια χρονοβόρα διαδικασία, με δυσκολία ανάλογη της πολυπλοκότητας του χωρίου. Μάλιστα, η κατασκευή του πλέγματος και του σωστού «κατάλληλου-προς-ανάλυση» γεωμετρικού μοντέλου καταλαμβάνει το 80% του συνολικού χρόνου που απαιτείται για τη μοντελοποίηση και την επίλυση ενός προβλήματος συνοριακών τιμών. [1] Αφ'ετέρου, η διαδικασία Ανάλυσης ενός προβλήματος μηχανικής αναφέρεται στο χώρο του εμπορίου ως *Μηχανική με τη βοήθεια Η/Υ* (Computer-aided engineering, CAE).

Στα περισσότερα συστήματα CAD η σχεδίαση βασίζεται σε συναρτήσεις βάσης spline και ιδιαίτερα στις μη ομοιόμορφες ρητές συναρτήσεις B-spline (Non-uniform rational B-splines, NURBS). Με τη μεταφορά του γεωμετρικού μοντέλου σε ένα περιβάλλον Ανάλυσης Πεπερασμένων Στοιχείων, το γεωμετρικό μοντέλο παρεμβάλλεται με τη χρήση τμηματικών πολυωνύμων Lagrange, με βάση τα οποία κατασκευάζεται ένα πλέγμα πεπερασμένων στοιχείων το οποίο προσεγγίζει, χωρίς να αναπαριστά ακριβώς, την αρχική ακριβή γεωμετρία CAD. Αυτή η προσέγγιση μπορεί σε πολλές περιπτώσεις να δημιουργήσει σφάλματα στα αναλυτικά αποτελέσματα.

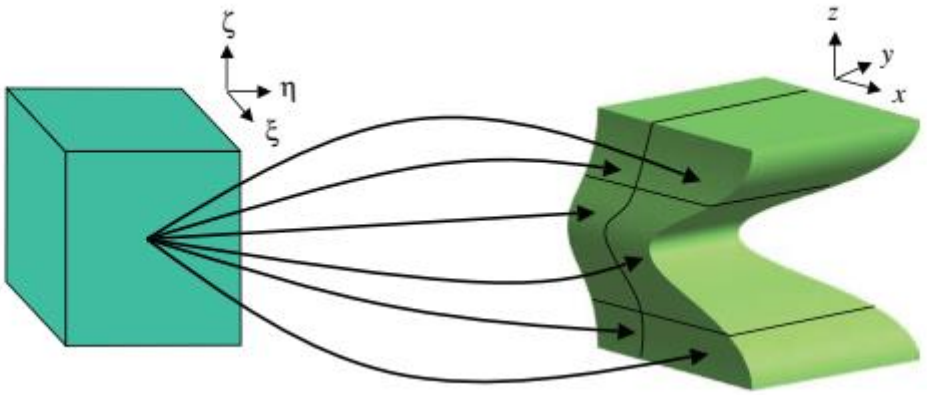
Με την πάροδο των ετών, δημιουργήθηκαν ολοκληρωμένα περιβάλλοντα CAD/CAE, τα οποία συγχρονίζουν τη μοντελοποίηση και την ανάλυση, διευκολύνοντας το έργου του Μηχανικού. Η ενσωμάτωση των δύο μεθοδολογιών, όμως, μπορεί να προχωρήσει και ένα βήμα παρακάτω, με την αξιοποίηση του γεωμετρικού μοντέλου της CAD ως μοντέλου ανάλυσης για την CAE. Αυτή η προσέγγιση απαιτεί την μετάβαση από μια Ανάλυση πεπερασμένων στοιχείων με χρήση των κλασικών πολυωνύμων παρεμβολής σε μια Ανάλυση που βασίζεται στις συναρτήσεις παρεμβολής της CAD. Η ρηξικέλευθη αυτή ιδέα αποτελεί τον πυρήνα της Ισογεωμετρικής Ανάλυσης και παρουσιάστηκε στο [1]. Η χρήση των ίδιων συναρτήσεων τόσο για την κατασκευή της γεωμετρίας όσο και για τον ορισμό της βάσης για το χώρο των λύσεων θεμελιώνεται στην ισοπαραμετρική αρχή, στην οποία βασίζεται και η κλασική Ανάλυση Πεπερασμένων Στοιχείων. Η διαφορά μεταξύ των δύο μεθόδων έγκειται στο γεγονός ότι στην Ισογεωμετρική Ανάλυση χρησιμοποιούμε τις γεωμετρικές συναρτήσεις (NURB/T-spline), με τις οποίες μπορούμε να αναπαραστήσουμε με ακρίβεια τη γεωμετρία, αποφεύγοντας, έτσι, τα σφάλματα που προκύπτουν από τα προσεγγιστικά γεωμετρικά μοντέλα της Ανάλυσης Πεπερασμένων Στοιχείων.

2 B-SPLINE, NURBS

2.1 B-spline

Οι συναρτήσεις B-spline είναι τμηματικά πολυώνυμα, με τα οποία επιτυγχάνεται η ακριβής περιγραφή μεγάλου πλήθους γεωμετρικών κατασκευών. Τα αντικείμενα (καμπύλες, επιφάνειες, όγκοι) B-spline κατασκευάζονται από το γραμμικό συνδυασμό των συναρτήσεων βάσης, οι οποίες παράγουν τον αντίστοιχο χώρο B-spline.

Όπως αναφέρεται στο [2], σε αντίθεση με την Ανάλυση Πεπερασμένων Στοιχείων, ο παραμετρικός χώρος (δηλαδή, ο χώρος, ο οποίος παράγεται από τις συναρτήσεις μορφής) είναι τοπικός ως προς ολόκληρο το μακρο-στοιχείο (macro-element, [3]) ή αλλιώς χωρίο κι όχι ως προς καθένα από τα επιμέρους στοιχεία (elements) που το αποτελούν. Πιο συγκεκριμένα, σημειώνεται ότι στην Ανάλυση Πεπερασμένων Στοιχείων, ο παραμετρικός χώρος, ο οποίος ταυτίζεται με το στοιχείο αναφοράς (ή γονικό στοιχείο), απεικονίζεται σε καθένα από τα στοιχεία του φυσικού χώρου και για κάθε στοιχείο του φυσικού χώρου ορίζεται μια ξεχωριστή γεωμετρική απεικόνιση (Σχήμα 2-1). Αντίθετα, στην Ισογεωμετρική Ανάλυση, κάθε χωρίο –το οποίο αποτελείται από ένα πλήθος στοιχείων–απεικονίζεται στο φυσικό χώρο, όπως στο Σχήμα 2-2. Δηλαδή, κάθε στοιχείο του φυσικού χώρου είναι η εικόνα του αντίστοιχου παραμετρικού στοιχείου και η γεωμετρική απεικόνιση που περιγράφει αυτή την αντιστοίχιση παραμένει ίδια για το σύνολο των παραμετρικών στοιχείων. Το *χωρίο* διαδραματίζει το ρόλο της υπο-περιοχής (subdomain) της γεωμετρίας, εντός της οποίας τα επιμέρους στοιχεία μοιράζονται ομοιόμορφες ιδιότητες ως προς τη φύση του γεωμετρικής κατασκευής. Πολλά απλά γεωμετρικά πεδία, αλλά και πιο σύνθετα [1] μπορούν να αναπαρασταθούν από ένα και μόνο χωρίο, καθώς και οι γεωμετρίες που θα μας απασχολήσουν στη συνέχεια.



Σχήμα 2-1 Στην κλασική ανάλυση πεπερασμένων στοιχείων, ο παραμετρικός χώρος είναι τοπικός ως προς τα μεμονωμένα στοιχεία Κάθε στοιχείο έχει τη δική του ξεχωριστή απεικόνιση από το γονικό στοιχείο (ή στοιχείο αναφοράς). [2]

2.2 Κομβοδιανύσματα και συναρτήσεις βάσης

Οι συναρτήσεις B-spline στη μια διάσταση υπολογίζονται με τη βοήθεια του *κομβοδιανύσματος* (knot vector). Το κομβοδιάνυσμα είναι μια μη φθίνουσα ακολουθία συντεταγμένων του

παραμετρικού χώρου που γράφεται στη μορφή $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$, όπου $\xi_i \in \mathbb{R}$ είναι ο i-οστός κόμβος, i είναι ο δείκτης του εκάστοτε κόμβου (knot index), p ο πολυωνυμικός βαθμός της βάσης και n το πλήθος των συναρτήσεων βάσης από τις οποίες ορίζεται η καμπύλη B-spline. Την ίδια συντεταγμένη στον παραμετρικό χώρο μπορεί να την μοιράζονται περισσότεροι από έναν κόμβοι ή ισοδύναμα, μια κομβική τιμή μπορεί να εμφανίζεται παραπάνω από μια φορά σε ένα κομβοδιάνυσμα. Οι κόμβοι διαμερίζουν το παραμετρικό χωρίο σε *στοιχεία* (elements). Το στοιχείο στη μία διάσταση καλείται, επίσης, *κομβοδιάστημα* (knot span), καθώς αποτελεί το διάστημα που εκτείνεται ανάμεσα σε δύο διαφορετικές τιμές κόμβων. Το σύνορο κάθε στοιχείου στο φυσικό χώρο είναι επί της ουσίας η εικόνα των κομβικών γραμμών μέσω της γεωμετρικής απεικόνισης που ορίζεται από τη βάση B-spline (Σχήμα 2-2). Ένα κομβοδιάνυσμα θα καλείται *ανοιχτό* (open knot vector), αν η πρώτη και η τελευταία κομβική τιμή έχουν πολλαπλότητα ίση με $p + 1$. Στην Ισογεωμετρική Ανάλυση χρησιμοποιούνται κατ' εξοχήν ανοιχτά κομβοδιανύσματα και εφεξής θα θεωρείται δεδομένο ότι κάθε κομβοδιάνυσμα είναι ανοιχτό, εκτός αν προσδιορίζεται διαφορετικά (π.χ. στις συναρτήσεις T-spline).

Οι συναρτήσεις B-spline βαθμού p ενός κομβοδιανύσματος Ξ ορίζονται στον παραμετρικό χώρο αναδρομικά, ξεκινώντας από τα τμηματικά σταθερά πολυώνυμα ($p = 0$)

$$N_{i,0}(\xi) = \begin{cases} 1, & \text{αν } \xi_i \leq \xi \leq \xi_{i+1} \\ 0, & \text{αλλιώς} \end{cases} \tag{2.1}$$

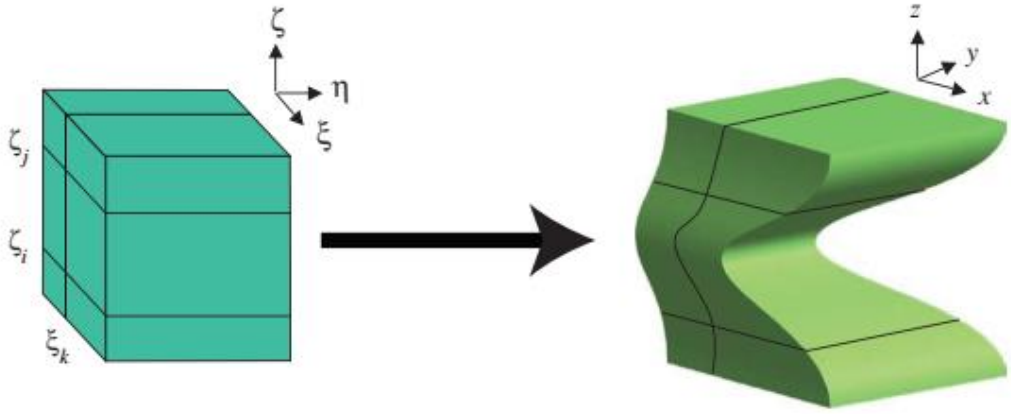
Για $p = 1,2,3, \dots$, οι συναρτήσεις βάσης ορίζονται ως

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{2.2}$$

Η παραπάνω σχέση καλείται *αναδρομική σχέση Cox-de Boor*.

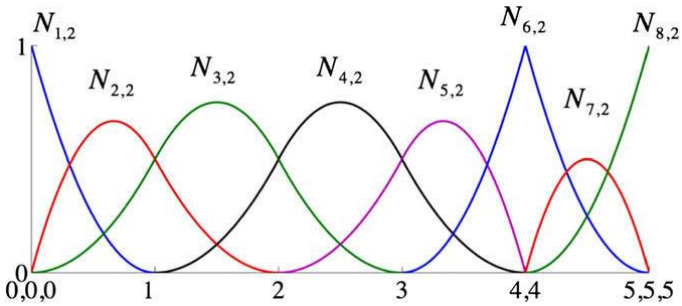
Οι συναρτήσεις B-spline ικανοποιούν τις ακόλουθες ιδιότητες: [4]

i. Διαμερίζουν τη μονάδα: $\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \xi \in [\xi_1, \xi_{n+p+1}]$.
ii. Είναι μη αρνητικές: $N_{i,p}(\xi) \geq 0, i = 1, \dots, n$.
iii. Είναι γραμμικά ανεξάρτητες: $\sum_{i=1}^{n} a_i N_{i,p}(\xi) = 0 \Leftrightarrow a_i = 0, i = 1, \dots, n$.
iv. Έχουν συμπαγή φορέα: $\{\xi | N_{i,p}(\xi) > 0\} \subset [\xi_i, \xi_{i+p+1}]$.
v. Έχουν ελεγχόμενη συνέχεια: Αν μια κομβική τιμή έχει πολλαπλότητα k (δηλαδή, $\xi_i = \xi_{i+1} = \cdots = \xi_{i+k-1}$), τότε οι συναρτήσεις βάσης σε αυτό το σημείο θα είναι C^{p-k}-συνεχείς. Αν $k = p$, οι συναρτήσεις βάσης έχουν μηδενική τάξη συνέχειας C^0 στη συγκεκριμένη θέση.



Σχήμα 2-2 Ο παραμετρικός χώρος B-spline είναι τοπικός ως προς ολόκληρο το μακρο-στοιχείο ή αλλιώς χωρίο. Οι εσωτερικοί κόμβοι διαμερίζουν το χωρίο σε στοιχεία. Μια απεικόνιση B-spline μεταφέρει το χωρίο από τον παραμετρικό χώρο στο φυσικό χώρο. [2]

Στο Σχήμα 2-3 απεικονίζεται ένα παράδειγμα δευτεροβάθμιας ($p = 2$) βάσης B-spline με $\Xi = \{0,0,0,1,2,3,4,4,5,5,5\}$, το οποίο παρουσιάστηκε στο [4]. Στις θέσεις της πρώτης και της τελευταίας κομβικής τιμής, η βάση είναι ασυνεχής στις τιμές 0 και 1 του πεδίου τιμών, λόγω του ανοιχτού κομβοδιανύσματος Ξ, ενώ στη συντεταγμένη $\xi = 4$, όπου η πολλαπλότητα της κομβικής τιμής είναι ίση με τον πολυωνυμικό βαθμό της βάσης ($p = 4$) η τάξη συνέχειας είναι C^0. Οπουδήποτε αλλού, οι συναρτήσεις έχουν τάξη συνέχειας C^1. Στις κομβικές τιμές, η τάξη συνέχειας της βάσης είναι C^{p-k}, όπου k η πολλαπλότητα της εκάστοτε κομβικής τιμής. Όταν η πολλαπλότητα είναι ίση με p, η βάση στο σημείο αυτό είναι απλώς συνεχής.



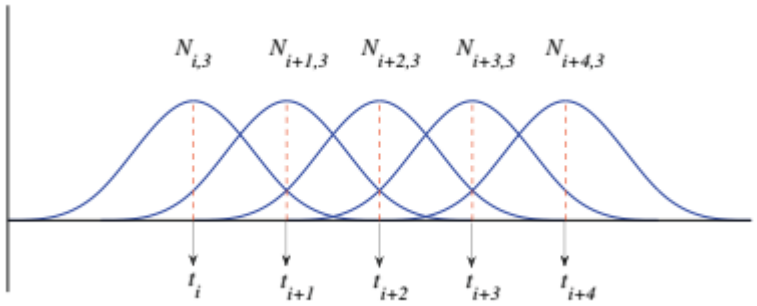
Σχήμα 2-3 Συναρτήσεις βάσης δευτέρου βαθμού (p = 2) για το ανοιχτό μη ομοιόμορφο κομβοδιάνυσμα $\Xi = \{0,0,0,1,2,3,4,4,5,5,5\}$. [4]

2.3 Δεσμοί

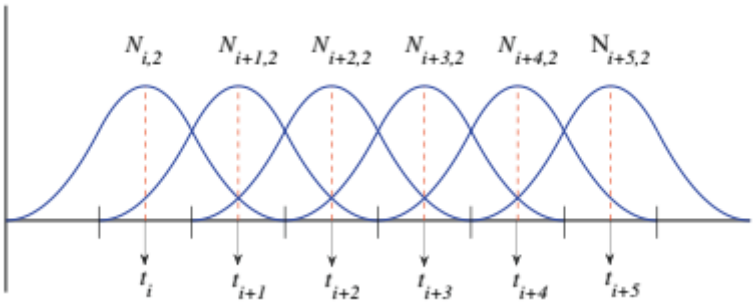
Οι κόμβοι $\{\xi_i\}_{i=1}^{n+p+1}$ του εκάστοτε κομβοδιανύσματος και οι συναρτήσεις βάσης $\{N_{j,p}\}_{j=1}^{n}$ που κατασκευάζονται από αυτό δε βρίσκονται σε ένα-προς-ένα αντιστοιχία. Δεδομένου ότι λαμβάνουμε υπόψιν μόνο ανοιχτά κομβοδιανύσματα, το πλήθος των κόμβων σε κάθε κομβοδιάνυσμα είναι $n + p + 1$, ενώ το πλήθος των συναρτήσεων που ορίζονται από αυτό είναι n. Εντούτοις, είναι ιδιαίτερα πρακτικό να προσδιορίσουμε θέσεις στο παραμετρικό χωρίο με τις οποίες θα σχετίζονται οι συναρτήσεις βάσης και στις οποίες θα αντιστοιχίζονται αμφιμονοσήμαντα. [4] Οι θέσεις αυτές καλούνται *δεσμοί* (anchors). Για κάθε συνάρτηση $N_{i,p}$, ο δεσμός της t_i ορίζεται ως

$$t_i = \begin{cases} \xi_{i+\frac{(p+1)}{2}}, & \text{αν } p \text{ περιττός} \\ \frac{1}{2}\left(\xi_{i+\frac{p}{2}} + \xi_{i+\frac{p}{2}+1}\right), & \text{αν } p \text{ άρτιος} \end{cases} \tag{2.3}$$

Η παραπάνω ιδέα απεικονίζεται για ένα ομοιόμορφο κομβοδιάνυσμα στο Σχήμα 2-4 και στο Σχήμα 2-5. Στην περίπτωση κόμβου[1] με πολλαπλότητα $k > 1$, δεσμοί που σχετίζονται με διαφορετικές συναρτήσεις βάσης μπορεί να έχουν την ίδια συντεταγμένη στον παραμετρικό χώρο.



Σχήμα 2-4 Βάση περιττού βαθμού, οι δεσμοί ταυτίζονται με τους κόμβους [4]



Σχήμα 2-5 Βάση άρτιου βαθμού, οι δεσμοί βρίσκονται στο κέντρο των κομβοδιαστημάτων [4]

2.4 Παράγωγοι συναρτήσεων B-spline

Με τη βοήθεια της μαθηματικής επαγωγής αποδεικνύεται ότι η παράγωγος μιας συνάρτησης B-spline δίνεται από την αναδρομική σχέση

$$N'_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{2.4}$$

Γενικότερα, εάν με $N_{i,p}^{(k)}$ συμβολίζεται η παράγωγος τάξης k της $N_{i,p}(\xi)$, επανειλημμένες παραγωγίσεις της σχέσης (2.4) δίνουν την παρακάτω γενική σχέση

[1] Για λόγους απλοποίησης της ορολογίας, πολλές φορές ο όρος «κομβική τιμή» αντικαθίσταται από τον όρο «κόμβος».

$$N_{i,p}^{(k)} = \frac{p!}{(p-k)!} \sum_{j=0}^{k} a_{k,j} \, N_{i+j,p-k} \tag{2.5}$$

όπου

$$\begin{aligned} a_{0,0} &= 1 \\ a_{k,0} &= \frac{a_{k-1,0}}{\xi_{i+p-k+1} - \xi_i} \\ a_{k,j} &= \frac{a_{k-1,j} - a_{k-1,j-1}}{\xi_{i+p+j-k+1} - \xi_i} \; j = 1, \dots, k-1 \\ a_{k,k} &= \frac{-a_{k-1,k-1}}{\xi_{i+p+1} - \xi_{i+k}} \end{aligned} \tag{2.6}$$

2.5 Καμπύλες B-spline

Έστω d_s το πλήθος των διαστάσεων του φυσικού χώρου. Μια καμπύλη B-spline στο χώρο $\mathbb{R}^{d_s}$ ορίζεται ως

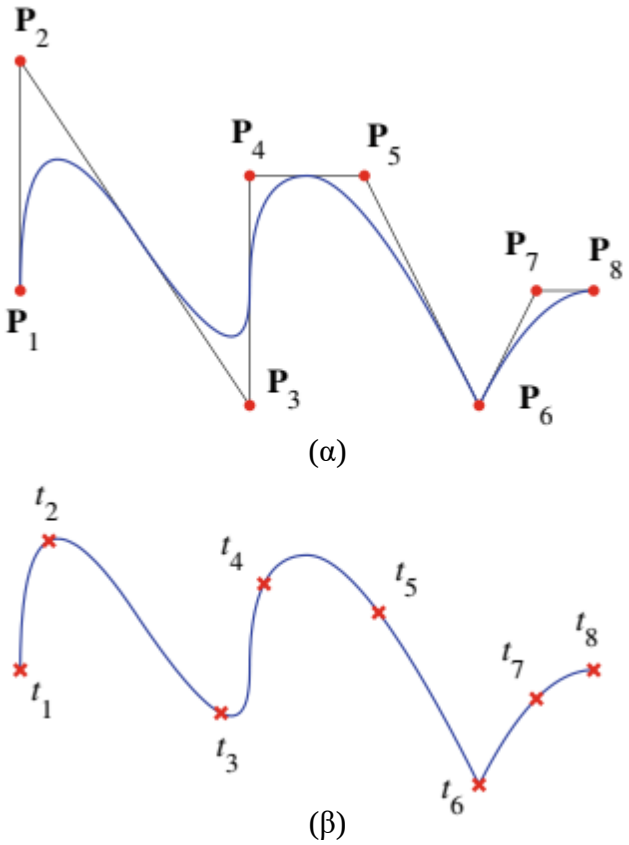
$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i N_{i,p}(\xi) \, , \tag{2.7}$$

όπου τα σημεία $\mathbf{P}_i \in \mathbb{R}^{d_s}, i = 1, \dots, n$ καλούνται *σημεία ελέγχου* (control points) της καμπύλης $\mathbf{C}$. Η τμηματικά γραμμική παρεμβολή των σημείων ελέγχου ορίζει το *πολύγωνο ελέγχου* (control polygon).

Οι καμπύλες B-spline διαθέτουν σημαντικές ιδιότητες, όπως [4]:

i. Αφινική συνδιακύμανση: Ένας αφινικός μετασχηματισμός της καμπύλης B-spline προκύπτει με την εφαρμογή του μετασχηματισμού κατευθείαν στα σημεία ελέγχου.
ii. Κυρτή θήκη: Μια καμπύλη B-spline βρίσκεται εντός της κυρτής θήκης των σημείων ελέγχου της.
iii. Μειωμένη διακύμανση (Variation diminishing): Κανένα επίπεδο δεν έχει περισσότερες τομές με την καμπύλη απ' όσες έχει με το πολύγωνο ελέγχου της.

Επιπλέον, οι καμπύλες B-spline κληρονομούν τις ιδιότητες συνέχειας των συναρτήσεων B-spline, από τις οποίες παράγονται, όπως φαίνεται και στο Σχήμα 2-6^{α}, όπου απεικονίζεται μια καμπύλη B-spline κατασκευασμένη από τις συναρτήσεις του Σχήμα 2-3. Στη θέση του φυσικού χώρου που αντιστοιχεί στην παραμετρική τιμή $\xi = 4$, η καμπύλη B-spline είναι απλώς συνεχής, ενώ το σημείο ελέγχου P_6 της καμπύλης είναι επίσης σημείο παρεμβολής. Η επιλογή ανοιχτών κομβοδιανυσμάτων εξασφαλίζει ότι το πρώτο και το τελευταίο σημείο ελέγχου, P_1 και P_8 είναι σημεία παρεμβολής. [4]



Σχήμα 2-6 Καμπύλη B-spline δευτέρου βαθμού στον $\mathbb{R}^2$. Το κομβοδιάνυσμα και οι συναρτήσεις βάσης της απεικονίζονται στο Σχήμα 2-3. (α) Η καμπύλη με τα σημεία ελέγχου και το πολύγωνο ελέγχου. Οι θέσεις των σημείων ελέγχου συμβολίζονται με κόκκινα ●. (β) Η καμπύλη με τους δεσμούς. Οι θέσεις των δεσμών συμβολίζονται με κόκκινα ×. [4]

Το πλήθος των δεσμών ταυτίζεται με το πλήθος των σημείων ελέγχου, καθιστώντας ευκολότερη τη συσχέτιση των σημείων ελέγχου με μία θέση στο χώρο δεικτών. Επίσης, τα σημεία ελέγχου είναι σε ένα-προς-ένα αντιστοιχία με τις συναρτήσεις βάσης. Επομένως, τα σημεία ελέγχου είναι σε ένα-προς-ένα αντιστοιχία και με τους δεσμούς. Όπως ο δεσμός μας πληροφορεί για τη θέση στην οποία βρίσκεται η εκάστοτε συνάρτηση βάσης στο παραμετρικό χωρίο, μας πληροφορεί, επίσης, και για τη θέση του εκάστοτε σημείου ελέγχου. (Σχήμα 2-6). [4]

2.6 Πολυδιάστατες συναρτήσεις B-spline

Οι πολυδιάστατες συναρτήσεις B-spline ορίζονται ως το τανυστικό γινόμενο των μονοδιάστατων συναρτήσεων B-spline. Συμβολίζουμε με d_p το σύνολο των μεταβλητών ή διαστάσεων και με $l = 1, \dots, d_p$ αριθμούμε κάθε διάσταση. Οι συναρτήσεις B-spline κατασκευάζονται από d_p κομβοδιανύσματα, δηλαδή σε κάθε κατεύθυνση αντιστοιχεί ένα κομβοδιάνυσμα. Γράφουμε

$$\Xi = \{\xi_1^l, \xi_2^l, \dots, \xi_{n_l+p_l+1}^l\} \tag{2.8}$$

όπου p_l ο πολυωνυμικός βαθμός της βάσης κατά τον παραμετρικό άξονα l και n_l το πλήθος των σχετιζόμενων συναρτήσεων βάσης. Οι εν λόγω μονοδιάστατες συναρτήσεις βάσης συμβολίζονται ως $\{N_{i_l,p_l}^l\}_{i_l}^{n_l}$ και ορίζονται από το κομβοδιάνυσμα Ξ, σύμφωνα με τις σχέσεις (2.1) και (2.2) .

Για τον ορισμό των πολυδιάστατων B-spline, χρησιμοποιείται ο πολύ-δείκτης $\mathbf{i} \in \mathbb{Z}^{d_p}$ και το σύνολο

$$I = \left\{\mathbf{i} = \left\{i_1, \dots, i_{d_p}\right\} \mid i_l \in \{1, \dots, n_l\}\ \forall l = 1, \dots, d_p\right\} \tag{2.9}$$

Ομοίως, το διάνυσμα των πολυωνυμικών βαθμών συμβολίζεται ως $\mathbf{p} = \left\{p_1, p_2, \dots, p_{d_p}\right\}$.

Έτσι, για κάθε πολύ-δείκτη $\mathbf{i} \in I$, ορίζεται η αντίστοιχη συνάρτηση B-spline διάστασης d_p ως

$$B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) = \prod_{l=1}^{d_p} N_{i_l,p_l}^{l}\left(\xi^l\right) \tag{2.10}$$

όπου $\boldsymbol{\xi} = \left(\xi^1, \xi^2, \dots, \xi^{d_p}\right)$.

Γίνεται σαφές από τον ορισμό ότι συνολικά υπάρχουν $\prod_{l=1}^{d_p} n_l$ συναρτήσεις βάσης, οι οποίες σχετίζονται με τα d_p κομβοδιανύσματα. Οι πολυδιάστατες συναρτήσεις B-spline κληρονομούν την πλειονότητα των ιδιοτήτων των μονοδιάστατων συναρτήσεων, όπως τη διαμέριση της μονάδας, τη μη αρνητικότητα, το συμπαγή φορέα, την υψηλή τάξη συνέχειας και τη γραμμική ανεξαρτησία. [4]

2.7 Επιφάνειες και όγκοι B-spline

Για τον ορισμό μιας επιφάνειας B-spline, απαιτείται ένα διάνυσμα πολυωνυμικών βαθμών $\mathbf{p} = \{p_1, p_2\}$, δύο κομβοδιανύσματα $\Xi^1 = \left\{\xi_1^1, \xi_2^1, \dots, \xi_{n_1+p_1+1}^1\right\}$ και $\Xi^2 = \left\{\xi_1^2, \xi_2^2, \dots, \xi_{n_2+p_2+1}^2\right\}$, ένα για κάθε κατεύθυνση και ένα σύνολο σημείων ελέγχου $\{\mathbf{P_i}\}_{\mathbf{i}\in I}$, το οποίο θα καλείται *πλέγμα ελέγχου* (control mesh) και αποτελεί το πολυδιάστατο ανάλογο του πολυγώνου ελέγχου (Κεφάλαιο 2.5).

Η επιφάνεια B-spline, ορίζεται, λοιπόν, ως

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} \mathbf{P_i} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) \tag{2.11}$$

όπου οι διδιάστατες συναρτήσεις βάσης $B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})$ ορίζονται από τη σχέση (2.10).

Με ανάλογο τρόπο, για να οριστεί ένας όγκος B-spline, απαιτείται ένα διάνυσμα πολυωνυμικών βαθμών $\mathbf{p} = \{p_1, p_2, p_3\}$, τρία κομβοδιανύσματα $\Xi^1 = \left\{\xi_1^1, \xi_2^1, \dots, \xi_{n_1+p_1+1}^1\right\}$, $\Xi^2 = \left\{\xi_1^2, \xi_2^2, \dots, \xi_{n_2+p_2+1}^2\right\}$ και $\Xi^3 = \left\{\xi_1^3, \xi_2^3, \dots, \xi_{n_2+p_2+1}^3\right\}$ και ένα πλέγμα ελέγχου $\{\mathbf{P_i}\}_{\mathbf{i}\in I}$. Ο όγκος B-spline ορίζεται, τότε, ως

$$\mathbf{V}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} \mathbf{P_i} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) \tag{2.12}$$

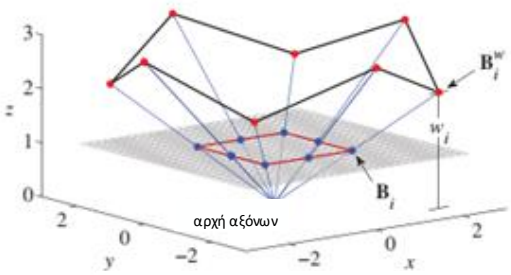
όπου $B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})$ τρισδιάστατες συναρτήσεις B-spline. Επισημαίνεται ότι συνηθίζεται να χρησιμοποιείται ο ίδιος πολυωνυμικός βαθμός p σε κάθε διάσταση, οπότε αναφερόμαστε στον πολυωνυμικό βαθμό της βάσης αντί στο διάνυσμα πολυωνυμικών βαθμών.

2.8 NURBS

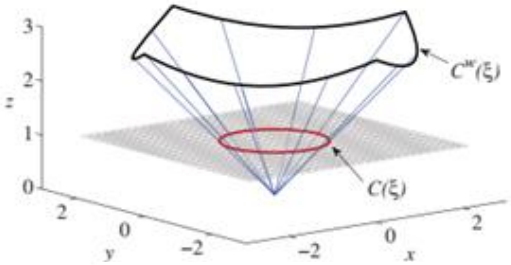
Οι μη ομοιόμορφες ρητές συναρτήσεις B-spline (Non-Uniform Rational B-Splines, NURBS) χρησιμοποιούνται κατά κόρον στη σχεδίαση γεωμετρικών αντικειμένων με την τεχνολογία CAD, καθώς μπορούν να αναπαραστήσουν γεωμετρικά αντικείμενα, όπως οι κωνικές τομές, τα οποία δεν θα μπορούσαν να περιγραφούν με ακρίβεια από τα τμηματικά πολυώνυμα B-spline. Η θεωρία των NURBS που ακολουθεί εξήχθη από το [2].

2.8.1 Γεωμετρική προσέγγιση

Ένα αντικείμενο NURBS στον χώρο $\mathbb{R}^d$ μπορεί να υπολογιστεί μέσω της προβολής ενός αντικειμένου B-spline του χώρου $\mathbb{R}^{d+1}$. Στο Σχήμα 2-7 απεικονίζεται ένα παράδειγμα κατασκευής κύκλου $\mathbf{C}(\xi)$ στον $\mathbb{R}^2$ από μια τμηματικά δευτεροβάθμια πολυωνυμική καμπύλη B-spline στον $\mathbb{R}^3$. Ο μετασχηματισμός υλοποιείται με την προβολή κάθε σημείου της καμπύλης B-spline πάνω στο επίπεδο $z = 1$ με κέντρο προβολής την αρχή των αξόνων O.



(α) Πολύγωνα ελέγχου



(β) Καμπύλες

Σχήμα 2-7 Ένας κύκλος στον $\mathbb{R}^2$ κατασκευάζεται από τον προβολικό μετασχηματισμό ενός τμηματικά δευτεροβάθμιου B-spline στον $\mathbb{R}^3$. (α) Ο προβολικός μετασχηματισμός του προβολικού σημείου ελέγχου B_i^w συνεπάγεται το σημείο ελέγχου B_i. Το βάρος w_i είναι η συντεταγμένη z του σημείου ελέγχου B_i^w. (β) Ο προβολικός μετασχηματισμός της καμπύλης B-spline $C^w(\xi)$ οδηγεί στην καμπύλη NURBS $C(\xi)$. [2]

Τα σημεία ελέγχου της καμπύλης NURBS υπολογίζονται εκτελώντας τον ίδιο προβολικό μετασχηματισμό στα σημεία ελέγχου της καμπύλης B-spline. Σε αυτό το πλαίσιο, η καμπύλη B-spline, $\mathbf{C}^w(\xi)$, καλείται *προβολική καμπύλη* και τα σημεία ελέγχου της, $\mathbf{P}_i^w$, *προβολικά σημεία ελέγχου*, ενώ οι όροι *καμπύλη* και *σημεία ελέγχου* προορίζονται για την καμπύλη NURBS $\mathbf{C}(\xi)$ και τα σημεία $\mathbf{P_i}$, αντίστοιχα.

Τα σημεία ελέγχου $\mathbf{P}_i$ δίνονται από τη σχέση

$$(\mathbf{P}_i)_j = \frac{(\mathbf{P}_i^w)_j}{w_i}, j = 1, \dots, d \tag{2.13}$$

με

$$w_i = (\mathbf{P}_i^w)_{d+1} \tag{2.14}$$

όπου $(\mathbf{P}_i)_j$, η j-οστή συντεταγμένη του διανύσματος $\mathbf{P}_i$ και w_i ο i-οστός συντελεστής βάρους. Τα βάρη w_i μπορούν να αναπαρασταθούν ως η συντεταγμένη z των προβολικών σημείων ελέγχου. Γενικότερα, συνιστούν την $d + 1$ συντεταγμένη των προβολικών σημείων ελέγχου στο χώρο $\mathbb{R}^{d+1}$ και απαιτείται στην πλειονότητα των υπολογιστικών εφαρμογών να είναι μη αρνητικοί αριθμοί. Η διαίρεση των προβολικών σημείων με τα βάρη ισοδυναμεί με την εφαρμογή σε αυτά του προβολικού μετασχηματισμού. Για την κατασκευή της ζητούμενης καμπύλης NURBS, χρειάζεται να εφαρμόσουμε τον ίδιο μετασχηματισμό σε κάθε σημείο της προβολικής καμπύλης, δηλαδή στο εν λόγω παράδειγμα, να διαιρέσουμε κάθε σημείο της καμπύλης B-spline στον $\mathbb{R}^3$ με το ύψος κάθε σημείου. Αυτό επιτυγχάνεται με τον ορισμό της *συνάρτησης βάρους* (weight function),

$$W(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) w_i \tag{2.15}$$

όπου $N_{i,p}(\xi)$ η συνάρτηση B-spline. Στον $\mathbb{R}^3$,η συνάρτηση $W(\xi) = z(\xi)$ αντιστοιχεί στο ύψος της καμπύλης συναρτήσει της παραμέτρου ξ. Δεδομένων όλων των παραπάνω, η καμπύλη NURBS μπορεί πλέον να οριστεί ως

$$\mathbf{C}(\xi)_j = \frac{\left(\mathbf{C}^w(\xi)\right)_j}{W(\xi)}, j = 1, \dots, d. \tag{2.16}$$

Καθώς οι συναρτήσεις $\mathbf{C}^w(\xi)$ και $W(\xi)$ είναι τμηματικά πολυώνυμα, η καμπύλη $\mathbf{C}(\xi)$ θα είναι μια τμηματικά ρητή συνάρτηση, δηλαδή εντός κάθε στοιχείου (ή αλλιώς, τμήματος) συνίσταται από ένα πηλίκο πολυωνύμων.

2.8.2 Αλγεβρική προσέγγιση

Οι συναρτήσεις NURBS, όπως και οι πολυωνυμικές B-spline, κατασκευάζονται από ένα κομβοδιάνυσμα. Θεωρούμε λοιπόν το κομβοδιάνυσμα $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$ με βαθμό p και $N_{i,p}$ τις n συναρτήσεις B-spline που κατασκευάζονται από αυτό. Με αφετηρία τις συναρτήσεις B-spline, $N_{i,p}$, για ένα δεδομένο σύνολο συντελεστών βάρους, w_i, μπορούμε να κατασκευάσουμε τις τμηματικά ρητές συναρτήσεις B-spline ως

$$R_i(\xi) = \frac{w_i \cdot N_{i,p}(\xi)}{W(\xi)}, \tag{2.17}$$

όπου

$$W(\xi) = \sum_{j=1}^{n} w_j \cdot N_{j,p}(\xi) \tag{2.18}$$

Ορίζοντας ως **W** το διαγώνιο πίνακα των συντελεστών βάρους,

$$\mathbf{W} = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_n \end{bmatrix} \tag{2.19}$$

και $\mathbf{N}(\xi)$ το διάνυσμα των συναρτήσεων B-spline, η εξίσωση (2.17) διατυπώνεται ισοδύναμα σε μορφή πινάκων ως

$$\mathbf{R}(\xi) = \frac{1}{W(\xi)} \mathbf{W}\mathbf{N}(\xi) \tag{2.20}$$

Η καμπύλη NURBS που ορίζεται από τις παραπάνω συναρτήσεις $R_{i,p}$ και τα σημεία ελέγχου $\mathbf{P} = \{\mathbf{P}_i\}_{i=1}^{n}$ εκφράζεται ως

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} R_{i,p}(\xi)\mathbf{P}_i \tag{2.21}$$

Οι πολυδιάστατες συναρτήσεις βάσης NURBS ορίζονται κατά ανάλογα τρόπο, χρησιμοποιώντας d_p κομβοδιανύσματα Ξ^l και ένα κατάλληλο σύνολο συντελεστών βάρους, $\{w_i\}_{i\in I}$, όπου I ένα κατάλληλο σύνολο δεικτών. Δεδομένων των **p** και **i**, οι αντίστοιχες συναρτήσεις βάσης NURBS ορίζονται ως

$$R_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) = \frac{w_{\mathbf{i}} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})}{\sum_{\mathbf{j}\in I} w_{\mathbf{j}} B_{\mathbf{j},\mathbf{p}}(\boldsymbol{\xi})} \tag{2.22}$$

Οι επιφάνειες και οι όγκοι NURBS ορίζονται όπως οι επιφάνειες και οι όγκοι B-spline, δηλαδή, ως εξής

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} R_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})\mathbf{P}_{\mathbf{i}} \tag{2.23}$$

Οι συναρτήσεις NURBS κληρονομούν όλες τις σημαντικές ιδιότητες των προκατόχων τους, των συναρτήσεων B-spline. Αυτές οι ιδιότητες περιλαμβάνουν τις εξής:

i. Διαμέριση της μονάδας,
ii. Μη αρνητικότητα σε κάθε σημείο,
iii. Αφινική συνδιακύμανση,
iv. Ιδιότητα της κυρτής θήκης,

ενώ η συνέχεια των NURBS συνεπάγεται με τον ίδιο τρόπο που προκύπτει στα B-spline. Επιπλέον, απαιτείται οι συντελεστές βάρους να είναι μη αρνητικοί αριθμοί, καθώς σε αντίθετη περίπτωση, μπορεί να παραβιάζεται η ιδιότητα της κυρτής θήκης. [4]

Όπως οι συναρτήσεις βάσης NURBS κατασκευάζονται από τις συναρτήσεις B-spline, έτσι και οι παράγωγοι των ρητών συναρτήσεων εξαρτώνται από τις παραγώγους των αντίστοιχων μη-ρητών συναρτήσεων. Συγκεκριμένα, έχουμε

$$\frac{d}{d\xi} R_i^p(\xi) = w_i \frac{W(\xi) N'_{i,p}(\xi) - W'(\xi) N_{i,p}(\xi)}{\left(W(\xi)\right)^2} \,, \tag{2.24}$$

όπου $N'_{i,p} \equiv \frac{d}{d\xi} N_{i,p}(\xi)$ και

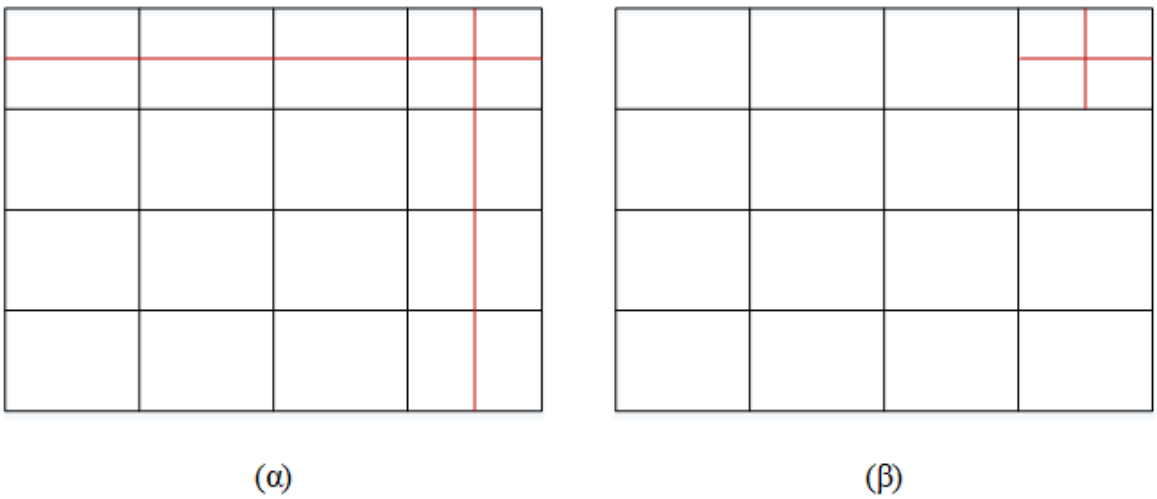
$$W'(\xi) = \sum_{k=1}^{n} N'_{k,p}(\xi) w_k \,. \tag{2.25}$$

3 T-SPLINE

3.1 Επισκόπηση

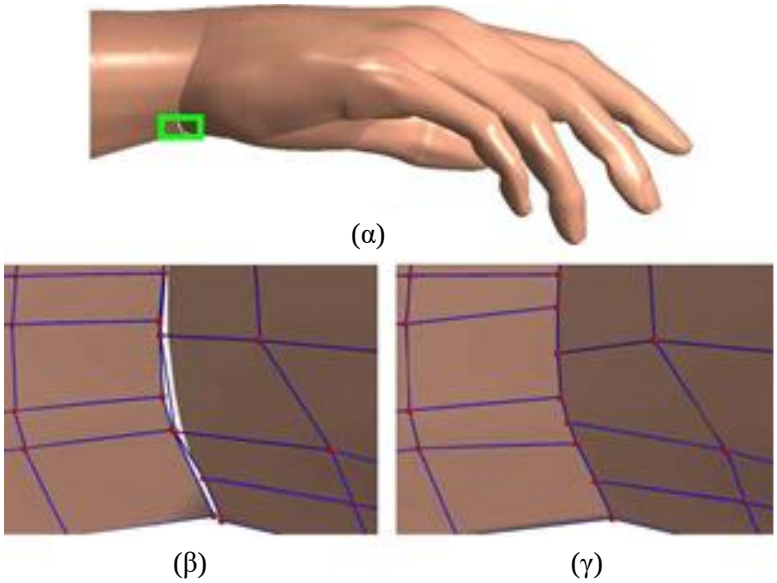
Οι συναρτήσεις T-spline εισήχθησαν το 2003 από τους Sederberg κ.ά. [5] ως γενίκευση της βάσης NURBS, με αφορμή τους περιορισμούς που προκύπτουν από την αυστηρή καρτεσιανή δομή των πολυδιάστατων NURBS. Τα T-spline, επί της ουσίας, συνδυάζουν την τοπολογία των NURBS με την ευελιξία των PB-spline. Ως PB-spline ορίζεται, για $d_p = 2$, η επιφάνεια της οποίας τα σημεία ελέγχου δεν έχουν τοπολογική σχέση μεταξύ τους κι η οποία κατασκευάζεται από μια συλλογή συναρτήσεων, τις λεγόμενες *συναρτήσεις ανάμειξης* (blending functions). [6] Κάθε συνάρτηση ανάμειξης είναι μια διδιάστατη συνάρτηση βάσης, η οποία παρήχθη από ένα ζεύγος *τοπικών κομβοδιανυσμάτων*, ενώ η χρήση του όρου "ανάμειξη" οφείλεται στην πιθανή γραμμική εξάρτηση των συναρτήσεων αυτών. Στα T-spline διατηρείται ο τοπικός χαρακτήρας των PB-spline με τη χρήση τοπικών κομβοδιανυσμάτων, αλλά σε αντίθεση με το πλέγμα των PB-spline, που βασίζεται σε ένα πλέγμα χωρίς ορισμένη τοπολογική δομή, επιλέγεται ένα πλέγμα παρόμοιο με εκείνο των NURBS, διατηρώντας, έτσι, πολλά από τα πλεονεκτήματα των συναρτήσεων NURBS. [6]

Μια επιφάνεια NURBS ορίζεται με τη χρήση σημείων ελέγχου, τα οποία βρίσκονται, τοπολογικά, εντός ενός τετράπλευρου πλέγματος, όπως στο Σχήμα 3-1^{α}. Αυτό συνεπάγεται ότι ένα πολύ μεγάλο ποσοστό των σημείων ελέγχου είναι περίσσιο, καθώς δεν περιέχει κάποια σημαντική γεωμετρική πληροφορία, παρά απαιτείται αποκλειστικά και μόνο για την ικανοποίηση των τοπολογικών περιορισμών του πλέγματος ελέγχου NURBS. Αντιθέτως, σε ένα πλέγμα ελέγχου T-spline επιτρέπεται να υπάρχουν ημιτελείς γραμμές σημείων ελέγχου που οδηγούν στο σχηματισμό διασταυρώσεων T. Αυτό το χαρακτηριστικό των T-spline συνεπάγεται μικρότερο αριθμό σημείων ελέγχου, άρα ο χρόνος που δαπανάται στη μοντελοποίηση της επιθυμητής γεωμετρίας είναι μικρότερος. Επιπλέον, η εκλέπτυνση του πλέγματος στα NURBS, δηλαδή η προσθήκη νέων σημείων ελέγχου στο πλέγμα χωρίς τη μεταβολή της επιφάνειας, συνεπάγεται την προσθήκη μιας ολόκληρης γραμμής σημείων ελέγχου, οδηγώντας στην καθολική εκλέπτυνση του πλέγματος. Αντίθετα, στα T-spline οι διασταυρώσεις T επιτρέπουν την τοπική εκλέπτυνση, παρ' ότι συνεχίζουν να υφίστανται περιορισμοί στην προσθήκη σημείων ελέγχου [7], [8].



Σχήμα 3-1 Εκλέπτυνση σε πλέγμα NURBS και T-spline (α) Καθολική εκλέπτυνση πλέγματος NURBS (β) Τοπική εκλέπτυνση πλέγματος T-spline

Ένας επιπλέον περιορισμός των NURBS είναι ότι καθώς κάθε επιφάνεια NURBS πρέπει να έχει ορθογώνια τοπολογία, πολλά γεωμετρικά αντικείμενα πρέπει να μοντελοποιηθούν με τη χρήση πολλών επιφανειών NURBS. Για την μοντελοποίηση του χεριού στο Σχήμα 3-2 χρησιμοποιήθηκαν επτά χωρία NURBS, ένα για τον πήχυ, ένα για την παλάμη και ένα για κάθε δάχτυλο, σύμφωνα με το [4]. Επιπλέον, αναφέρεται ότι η ομαλή συνένωση των επιφανείων σε μια κλειστή και ομαλή επιφάνεια NURBS ήταν αναποτελεσματική, όπως φαίνεται στο Σχήμα 3-2$^{\alpha}$ και στο Σχήμα 3-2$^{\beta}$. Παρ' όλα αυτά, ήταν εφικτή η συνένωση των επιφανειών NURBS σε μια επιφάνεια T-spline, χωρίς κενά, όπως φαίνεται και το Σχήμα 3-2$^{\gamma}$.



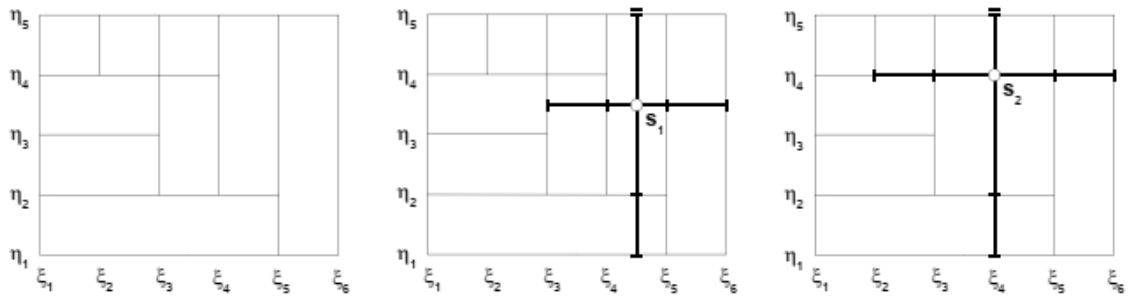
Σχήμα 3-2 (α) Μοντέλο χεριού κατασκευασμένο από επτά επιφάνειες NURBS (β) Μεγέθυνση της επισημασμένης περιοχής όπου απεικονίζεται το πλέγμα NURBS (γ) Το πλέγμα ελέγχου T-spline, όπου απεικονίζεται η συνένωση T-spline του πήχη και της παλάμης. [4]

Η βάση NURBS μπορεί να ιδωθεί ως μια υπο-περίπτωση βάσης T-spline, καθώς κάθε επιφάνεια NURBS είναι και επιφάνεια T-spline και μπορεί να μελετηθεί ως τέτοια. Σημειώνεται, δε, ότι η έννοια των διασταυρώσεων T σε ένα T-spline υπονοεί ότι το πλέγμα του T-spline έχει τουλάχιστον δύο διαστάσεις, δηλαδή ότι πρόκειται είτε για επιφάνεια είτε για στερεό. Ως εκ τούτου, θεωρείται τετριμμένη η περίπτωση ενός T-spline μιας διάστασης, δηλαδή μιας καμπύλης T-spline και δεν λαμβάνεται υπόψιν. Συγκεκριμένα, θα μελετηθούν διδιάστατα αντικείμενα T-spline, δηλαδή επιφάνειες T-spline στον τρισδιάστατο φυσικό χώρο $\mathbb{R}^3$.

3.2 Πλέγμα T-spline και τοπικά κομβοδιανύσματα

Στη βάση NURBS, οι συναρτήσεις βάσης ορίζονται στον παραμετρικό χώρο και ο χώρος δεικτών χρησιμοποιείται ως βοηθητικό εργαλείο. Στις συναρτήσεις T-spline, όμως, δίνεται μεγαλύτερη έμφαση στο χώρο δεικτών, καθώς σε αυτό τον χώρο ορίζεται αρχικά το πλέγμα T-spline ή αλλιώς T-mesh, το οποίο σε αυτή την περίπτωση καλείται T-mesh δεικτών (index T-mesh). Το T-mesh είναι ένα πολύ σημαντικό εργαλείο με το οποίο μπορούμε να ορίσουμε τόσο τις συναρτήσεις ανάμειξης όσο και τη διάταξη που έχουν μεταξύ τους. Ακολουθώντας τον ορισμό που δόθηκε στο [9], για ένα διδιάστατο T-spline βαθμού (d_1, d_2), το T-mesh είναι μια ορθογώνια διαμέριση του χώρου δεικτών $[0, c + d_1] \times [0, r + d_2]$, σύμφωνα με την οποία κάθε γωνία (ή αλλιώς, κορυφή) των επιμέρους

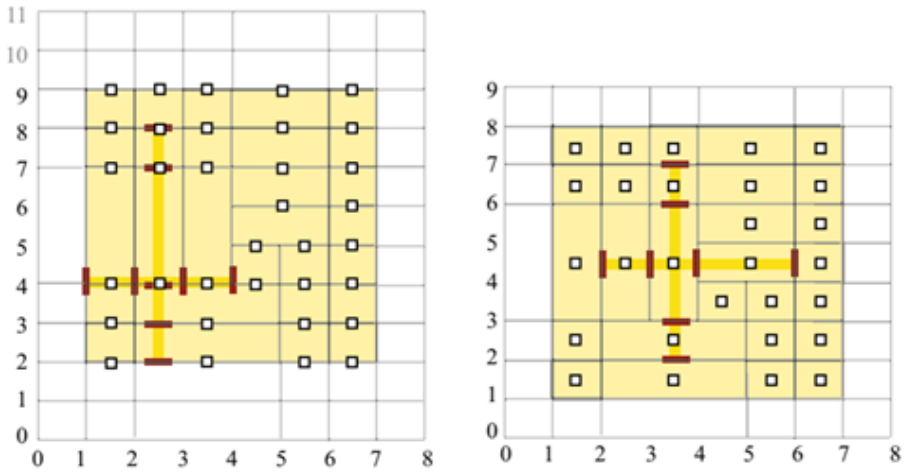
ορθογωνίων έχει ακέραιες συντεταγμένες και κάθε ορθογώνιο είναι ένα ανοιχτό σύνολο. Κάθε κορυφή, λοιπόν, αναπαρίσταται από ένα μοναδικό ζεύγος συντεταγμένων (δ_i, τ_i), όπου $\delta_i \in \mathbb{Z}$ και $\tau_i \in \mathbb{Z}$. [9] Το ευθύγραμμο τμήμα, το οποίο ενώνει δυο κορυφές στο T-mesh, χωρίς να παρεμβάλλεται ανάμεσα τους άλλη κορυφή, ονομάζεται *ακμή* (edge). Επιπλέον, η ορθογώνια περιοχή, στο εσωτερικό της οποίας δεν βρίσκεται καμία ακμή ή κορυφή καλείται *έδρα* (face). Το *σθένος* (valence) μιας κορυφής του T-mesh αντιστοιχεί στο πλήθος των ακμών με τις οποίες εκείνη ενώνεται. Στο εν λόγω πλαίσιο, επιτρέπουμε το σθένος για τις εσωτερικές κορυφές να είναι μόνο τρία (διασταύρωση σε σχήμα T, T-junction) και τέσσερα– δηλαδή, αποκλείονται πλέγματα με *εξαιρετικά σημεία* (extraordinary points). Οι πιθανές, λοιπόν, διασταυρώσεις T είναι της μορφής ⊢, ⊤, ⊥ και ⊣. Οι διασταυρώσεις T αντιστοιχούν στους κρεμαστούς «κόμβους» (hanging nodes) της Ανάλυσης Πεπερασμένων Στοιχείων. Επιπλέον, στο πλαίσιο της Ισογεωμετρικής Ανάλυσης, οι έδρες στο T-mesh καλούνται *στοιχεία του T-mesh*. Στο Σχήμα 3-3^{α} απεικονίζεται ένα απλό T-mesh.



(α) T-mesh (β) Δεσμός, $p = 2$ (γ) Δεσμός, $p = 3$

Σχήμα 3-3 T-mesh, δεσμοί για άρτιους και περιττούς πολυωνυμικούς βαθμούς. [6]

Οι *δεσμοί* (anchors) είναι σημεία στο T-mesh δεικτών, σε καθένα εκ των οποίων αντιστοιχίζεται από μια συνάρτηση ανάμειξης. Έστω ένα διδιάστατο T-spline βαθμού (d_1, d_2). Αν το d_1 και το d_2 είναι περιττοί αριθμοί, τότε οι δεσμοί ταυτίζονται με τις κορυφές του T-mesh, ενώ αν τα d_1, d_2 είναι άρτιοι, οι δεσμοί βρίσκονται στο κέντρο βάρους των εδρών του T-mesh. Έτσι, η συντεταγμένη του δεσμού είναι η συντεταγμένη της κάτω-αριστερής κορυφής της έδρας στην οποία βρίσκεται ο εν λόγω δεσμός. Στην περίπτωση που το d_1 είναι άρτιος και ο d_2 περιττός ή αντίστροφα, τότε ο δεσμός βρίσκεται στη μέση της οριζόντιας, ή αντίστοιχα κάθετης, ακμής στο T-mesh. Τότε, η συντεταγμένη του δεσμού είναι συντεταγμένη της αριστερής, ή αντίστοιχα κάτω, κορυφής της εν λόγω ακμής. Στο συγκεκριμένο πλαίσιο, μελετώνται T-spline, τα οποία έχουν τον ίδιο πολυωνυμικό βαθμό και ως προς τις δυο μεταβλητές και ο πολυωνυμικός βαθμός τους καλείται *(πολυωνυμικός) βαθμός του T-spline*.



(α) T-spline βαθμού (2,3) (β) T-spline βαθμού (2,2)

Σχήμα 3-4 Οι δεσμοί και η κατασκευή των τοπικών κομβοδιανυσμάτων στο χώρο δεικτών. [10]

Ένα έγκυρο T-mesh ορίζει μία συνάρτηση βάσης σε κάθε δεσμό. Έτσι, για κάθε δεσμό $\mathbf{s}_a = \{i, j\}$ του πλέγματος T-spline υπάρχει μια συνάρτηση ανάμειξης T-spline, η οποία χαρακτηρίζεται από ένα *τοπικό κομβοδιάνυσμα* (local knot vector) Ξ_a^i για κάθε κατεύθυνση i.

Η κατασκευή του τοπικού κομβοδιανύσματος πραγματοποιείται ως εξής [4]: Στην αρχή, θεωρούμε ότι το κομβοδιάνυσμα είναι ένα κενό διάνυσμα. Στη συνέχεια, παίρνουμε τις συντεταγμένες του δεσμού $\mathbf{s}_a = \{i, j\}$ και τοποθετούμε τις τιμές ξ_i^1 και ξ_j^2 στο κομβοδιάνυσμα Ξ_a^1 και στο Ξ_a^2, αντίστοιχα. Οι τιμές αυτές θα παραμείνουν στο κέντρο των τοπικών κομβοδιανυσμάτων τα οποία θα σχηματιστούν για κάθε κατεύθυνση. Έπειτα, εργαζόμαστε σε κάθε κατεύθυνση, ξεχωριστά, ξεκινώντας από την κατασκευή του κομβοδιανύσματος Ξ_a^1. Διασχίζουμε τον οριζόντιο άξονα στα δεξιά του δεσμού, καταγράφουμε την συντεταγμένη k της πρώτης κάθετης ακμής που θα συναντήσουμε και τοποθετούμε την τιμή ξ_k^1 στο τέλος του τοπικού κομβοδιανύσματος Ξ_a^1. Συνεχίζουμε αυτή τη διαδικασία μέχρι να συναντήσουμε στα δεξιά του δεσμού $(p + 1)$ / 2 κάθετες ακμές. Έπειτα, κατευθυνόμαστε πάλι οριζόντια, στα αριστερά του δεσμού, όμως, αυτή τη φορά, καταγράφουμε την συντεταγμένη k της πρώτης κάθετης ακμής που θα συναντήσουμε και τοποθετούμε την τιμή ξ_k^1 στην αρχή του τοπικού κομβοδιανύσματος Ξ_a^1. Συνεχίζουμε αυτή τη διαδικασία μέχρι να συναντήσουμε στα αριστερά του δεσμού $(p + 1)$ / 2 κάθετες ακμές. Οι κόμβοι του τοπικού κομβοδιανύσματος για την δεύτερη κατεύθυνση υπολογίζονται με παρόμοιο τρόπο. Καταγράφουμε τους πρώτους $(p + 1)$ / 2 κόμβους των κάθετων, ως προς την κάθετο άξονα, ακμών που συναντάμε πάνω και κάτω από τον δεσμό και τους τοποθετούμε στις αντίστοιχες θέσεις του κομβοδιανύσματος Ξ_a^2. Αν καθώς διασχίζουμε κάποιον άξονα, δεν υπάρχουν άλλες κάθετες ακμές, οι οποίες τον τέμνουν, αλλά υπάρχουν κενές θέσεις στο κομβοδιάνυσμα, συμπληρώνουμε τις θέσεις αυτές με την τιμή του τελευταίου καταγεγραμμένου κόμβου.

Έστω, για παράδειγμα, ότι μελετάται το T-mesh βαθμού $p = 2$, που παρουσιάζεται στο [6] και το οποίο απεικονίζεται στο Σχήμα 3-3$^{\beta}$. Σύμφωνα με τα παραπάνω, τα τοπικά κομβοδιανύσματα για το δεσμό $s_1 = ((\xi_4 + \xi_5)/2\,, (\eta_3, +\eta_4)/2)$ είναι τα $\Xi_1 = \{\xi_3, \xi_4, \xi_5, \xi_6\}$ και $H_1 = \{\eta_1, \eta_2, \eta_5, \eta_5\}$ και το μήκος τους είναι ίσο με $(p + 2) = 4$. Αν ο πολυωνυμικός βαθμός του T-spline είναι περιττός, όπως συμβαίνει στο Σχήμα 3-3$^{\gamma}$, όπου ο πολυωνυμικός βαθμός είναι $p = 3$. Το μήκος κάθε

κομβοδιανύσματος θα είναι και πάλι $p+2$, δηλαδή για $p=3$, $|\Xi_i|=|H_i|=5$. Τα τοπικά κομβοδιανύσματα για το δεσμό $s_2=(\xi_4,\eta_4)$ είναι τα $\Xi_2=\{\xi_2,\xi_3,\xi_4,\xi_5,\xi_6\}$ και $H_2=\{\eta_1,\eta_2,\eta_4,\eta_5,\eta_5\}$.

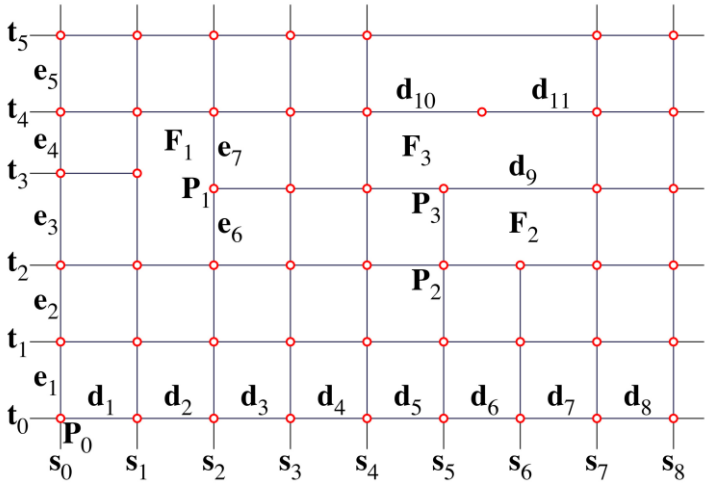
Στο Σχήμα 3-4 απεικονίζεται η κατασκευή των κομβοδιανυσμάτων στο ίδιο πλέγμα T-spline, αλλά με διαφορετικό πολυωνυμικό βαθμό ανά κατεύθυνση. Γίνεται σαφές, λοιπόν, ότι η τοπολογία ενός πλέγματος T-spline δεν καθορίζει εκ των προτέρων ούτε τις θέσεις ούτε το πλήθος των δεσμών του πλέγματος.

Σε πλείστες βιβλιογραφικές αναφορές ([11, 12, 5, 7, 13]), στην περίπτωση T-spline περιττού βαθμού, το T-mesh ορίζεται και ως το πλέγμα ελέγχου, καθώς τα σημεία ελέγχου ταυτίζονται με τις κορυφές (ή ισοδύναμα, δεσμούς) του T-mesh του χώρου δεικτών. Τότε, οι ακμές στο πλέγμα ελέγχου ταυτίζονται με τα διαστήματα μεταξύ των κόμβων. Επομένως, μπορεί να ανατεθεί *μια έγκυρη διάταξη κομβικών διαστημάτων* (knot interval configuration) στο πλέγμα ελέγχου, χωρίς να χρειάζεται να υπολογιστούν προηγουμένως τα τοπικά κομβοδιανύσματα. Για την έγκυρη διάταξη των κομβικών διαστημάτων στο T-mesh πρέπει να πληρούνται οι δυο παρακάτω προϋποθέσεις [5]:

Κανόνας 1. Το άθροισμα του μήκους των κομβικών διαστημάτων στις απέναντι ακμές κάθε έδρας πρέπει να είναι ίδιο.

Κανόνας 2. Αν μια διασταύρωση T στη μια ακμή μιας έδρας μπορεί να ενωθεί με τη διασταύρωση T σε μια απέναντι ακμή της έδρας (δηλαδή, αν μπορεί να χωριστεί η έδρα σε δύο έδρες), τότε χωρίς την παραβίαση του Κανόνα 1, η ακμή αυτή θα πρέπει να συμπεριληφθεί στο T-mesh.

Όπως αναφέρεται στο [11], με την έγκυρη διάταξη των κομβικών διαστημάτων, δημιουργείται ένα *σύστημα κομβικών συντεταγμένων* (knot coordinate system), με βάση το οποίο ορίζεται στη συνέχεια η επιφάνεια T-spline. Για την κατασκευή ενός συστήματος κομβικών συντεταγμένων, επιλέγεται ένα σημείο ελέγχου, του οποίου η προ-εικόνα χρησιμεύει ως αρχή για το παραμετρικό χωρίο $(s,t)=(0,0)$. Για παράδειγμα, στο Σχήμα 3-5, το σημείο (s_0,t_0) ορίζεται ως το σημείο αρχής του συστήματος συντεταγμένων. Μόλις επιλεγεί το σημείο αρχής, ανατίθεται μια κομβική τιμή s σε κάθε κάθετη ακμή της τοπολογίας του T-mesh και μια κομβική τιμή t σε κάθε οριζόντια ακμή της τοπολογίας του T-mesh. Στο Σχήμα 3-5, αυτές οι κομβικές τιμές συμβολίζονται ως s_i και t_i. Δεδομένης της αρχής των κόμβων, έχουμε $s_0=t_0=0$, $s_1=d_1$, $s_2=d_1+d_2$, $s_3=d_1+d_2+d_3$, $t_1=e_1$, $t_2=e_1+e_2$ και ούτω καθεξής. Ομοίως, κάθε σημείο ελέγχου θα έχει κομβικές συντεταγμένες. Για παράδειγμα, οι κομβικές συντεταγμένες για το σημείο P_0 είναι (0,0), για το P_1 είναι (s_2,t_2+e_6), για το P_2 είναι (s_5,t_2) και για το P_3 είναι (s_5,t_2+e_6). [7]



Σχήμα 3-5 Προ-εικόνα ενός T-mesh [7]

3.3 Διανύσματα τοπικών κομβοδιαστημάτων

Συχνά δεν έχει ιδιαίτερη σημασία ποιος είναι ο πρώτος κόμβος του τοπικού κομβοδιανύσματος, αλλά η απόσταση μεταξύ των κόμβων του. Έτσι, ορίζεται το *τοπικό διάνυσμα κομβοδιαστημάτων* (local knot interval vector) ως η ακολουθία των κομβοδιαστημάτων, $\Delta\Xi = \{\Delta\xi_1, \Delta\xi_2, \dots, \Delta\xi_{p+1}\}$, όπου $\Delta\xi_i = \xi_{i+1} - \xi_i$. Σύμφωνα με το [11], το *τοπικό πεδίο συνάρτησης βάσης* (local basis function domain) $\widehat{\Omega}_A \subset \mathbb{R}^{d_p}$ ορίζεται τότε ως $\widehat{\Omega}_A = [0, \Delta\xi_1 + \Delta\xi_2 + \cdots + \Delta\xi_{p+1}] \otimes [0, \Delta\eta_1 + \Delta\eta_2 + \cdots + \Delta\eta_{p+1}]$, $A = 1,2,\dots,n$, όπου n είναι το πλήθος των σημείων ελέγχου. Επιπλέον, σημειώνεται, ότι κάθε τοπικό πεδίο φέρει ένα τοπικό σύστημα συντεταγμένων $\boldsymbol{\xi}_A = (\xi_A^1, \xi_A^2) = (\xi_A, \eta_A)$, το οποίο καλείται *σύστημα συντεταγμένων της βάσης* (basis coordinate system).

Στο παράδειγμα του Σχήμα 3-3, αν $\xi_1 = 1, \xi_2 = 2, \dots, \xi_6 = 6$ και $\eta_1 = 1, \eta_2 = 2, \dots \eta_5 = 5$, τότε τα τοπικά διανύσματα κομβοδιαστημάτων στο s_1 είναι τα $\Delta\Xi_1 = \{1,1,1\}$ και $\Delta H_1 = \{1,3,0\}$ και για το s_2 είναι τα $\Delta\Xi_2 = \{1,1,1,1\}$ και $\Delta H_2 = \{1,2,1,0\}$. [6]

3.4 Συναρτήσεις ανάμειξης T-spline

Έστω ένα πλέγμα T-spline που αποτελείται από n δεσμούς. Κάθε δεσμός a αντιστοιχίζεται σε μία πολυδιάστατη συνάρτηση ανάμειξης N_a. Κάθε πολυδιάστατη συνάρτηση ανάμειξης N_a ορίζεται ως

$$N_a(\boldsymbol{\xi}) = \prod_{l=1}^{d} N_a^l(\xi_l)\,, \tag{3.1}$$

με N_a^l η μονοδιάστατη συνάρτηση B-spline βαθμού p_l για το δεσμό i και για την κατεύθυνση l, όπως ορίζεται σύμφωνα με τον αναδρομικό τύπο Cox-de Boor (2.1) και (2.2).

Η επιφάνεια T-spline, όπως και οι επιφάνειες B-spline και NURBS, ορίζεται ως ο γραμμικός συνδυασμός των συναρτήσεων βάσης και των σημείων ελέγχου της επιφάνειας, δηλαδή ως

$$T(\boldsymbol{\xi}) = \frac{\sum_{a=1}^{n} w_a N_a(\boldsymbol{\xi}) \mathbf{P}_a}{\sum_{a=1}^{n} w_a N_a(\boldsymbol{\xi})}, \tag{3.2}$$

όπου $\mathbf{P}_a$ είναι τα σημεία ελέγχου, w_a οι συντελεστές βάρους και N_a οι συναρτήσεις ανάμειξης. Ισοδύναμα, η επιφάνεια T-spline μπορεί να διατυπωθεί ως

$$T(\boldsymbol{\xi}) = \sum_{a=1}^{n} \frac{w_a N_a(\boldsymbol{\xi})}{\sum_{b=1}^{n} w_b N_b(\boldsymbol{\xi})} \mathbf{P}_a = \sum_{a=1}^{n} R_a(\boldsymbol{\xi}) \mathbf{P}_a\,, \tag{3.3}$$

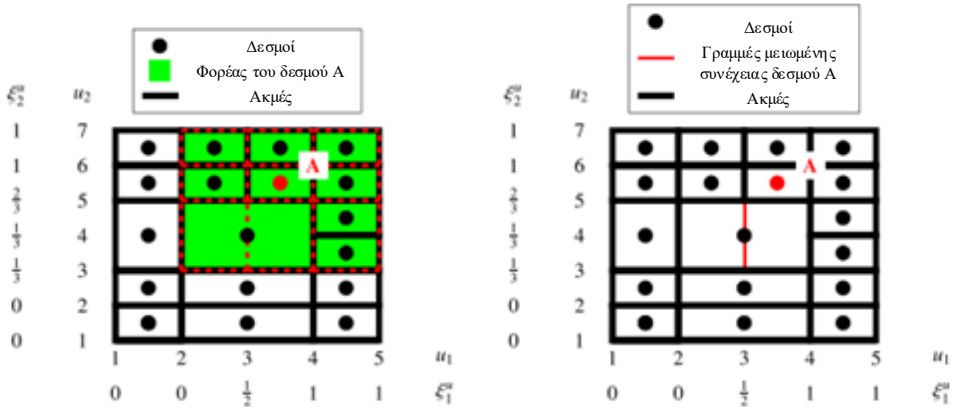
όπου $R_a(\boldsymbol{\xi}), a = 1, \dots n$ οι ρητές συναρτήσεις ανάμειξης T-spline, οι οποίες υπολογίζονται σύμφωνα με την εξίσωση (2.20). Είναι προφανές ότι οι ρητές συναρτήσεις R_a αθροίζουν πάντα στη μονάδα, ανεξάρτητα από την επιλογή των w_a. Για την ικανοποίηση της αφινικής συνδιακύμανσης, αρκεί οι συναρτήσεις R_a να διαμερίζουν τη μονάδα και όχι απαραίτητα οι πολυωνυμικές συναρτήσεις N_a.

3.5 Χώρος T-spline

Στο [7] ορίζεται ο χώρος T-spline ως το σύνολο όλων των T-spline τα οποία μοιράζονται την ίδια τοπολογία στο T-mesh, τα ίδια κομβοδιαστήματα και το ίδιο σύστημα κομβικών συντεταγμένων. Επιπλέον σημειώνεται ότι ένας χώρος T-spline μπορεί να αναπαρασταθεί με το διάγραμμα μιας προ-εικόνας (pre-image) ενός T-mesh, όπως στο Σχήμα 3-5. Δεδομένου ότι όλα τα T-spline σε ένα χώρο T-spline έχουν την ίδια προ-εικόνα, χαρακτηρίζεται ορθότερο να μιλάμε για την προ-εικόνα ενός χώρου T-spline.

3.6 Στοιχεία T-spline & Στοιχειώδες T-mesh

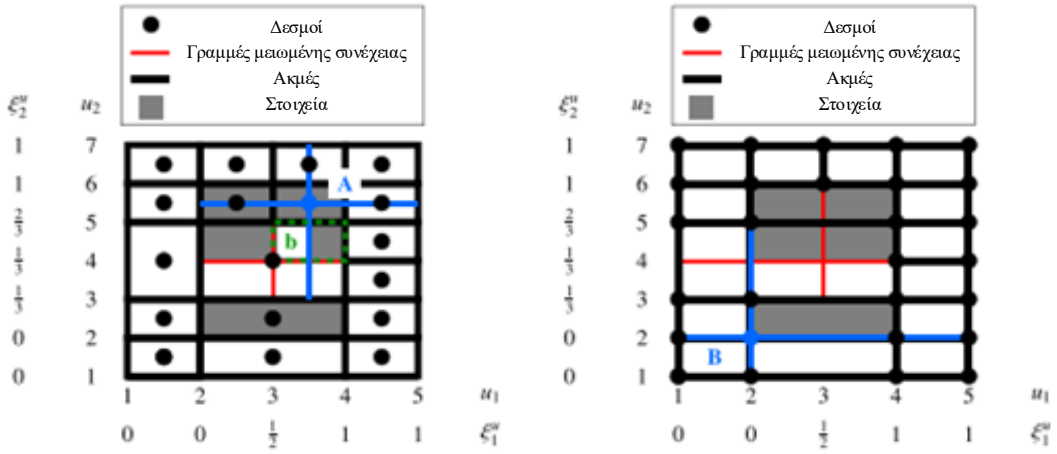
Σε ένα πλέγμα NURBS, οι γραμμές στις οποίες οι συναρτήσεις εμφανίζουν μειωμένη συνέχεια ταυτίζονται με τις γραμμές των κόμβων. Αντίθετα, σε ένα πλέγμα T-spline, λόγω των «ημιτελών» κομβικών γραμμών, υπάρχουν επιπλέον γραμμές μειωμένης συνέχειας, οι οποίες προεκτείνουν τις κομβικές γραμμές στις διασταυρώσεις T και συνήθως συμβολίζονται με διακεκομμένες γραμμές. Στο Σχήμα 3-6 απεικονίζεται η διαδικασία σχηματισμού των γραμμών μειωμένης συνέχειας για το δεσμό A σε ένα δευτεροβάθμιο T-spline, όπως παρουσιάστηκε στο [14]. Ο δεσμός A έχει συντεταγμένες $(3.5, 5.5)$ στο χώρο δεικτών και τοπικά κομβοδιανύσματα $\Xi_1^A = \{\xi_1^2, \xi_1^3, \xi_1^4, \xi_1^5\}$ και $\Xi_2^A = \{\xi_2^3, \xi_2^5, \xi_2^6, \xi_2^7\}$. Ο φορέας του δεσμού A έχει φορέα στην πράσινη υπο-περιοχή $[\xi_1^2, \xi_1^5] \times [\xi_2^3, \xi_2^7]$ του παραμετρικού χωρίου. Εντός αυτής της περιοχής, η σχεδίαση όλων των τιμών που περιέχονται στα τοπικά κομβοδιανύσματα Ξ_i^A οδηγεί στο πλέγμα των κόκκινων γραμμών του Σχήμα 3-6$^{\alpha}$. Κατά μήκος αυτών των γραμμών, οι συναρτήσεις ανά κατεύθυνση εμφανίζουν μειωμένη συνέχεια, ως αποτέλεσμα της πολλαπλότητας στις τιμές των κόμβων στα τοπικά κομβοδιανύσματα κάθε κατεύθυνσης. Αν κάποια από αυτές τις γραμμές δεν είναι και ακμή του πλέγματος T-spline, τότε προστίθεται στο πλέγμα, Σχήμα 3-6$^{\beta}$. Η γραμμή αυτή θα καλείται *γραμμή μειωμένης συνέχειας* (line of reduced continuity).



(α) (β)

Σχήμα 3-6 Γραμμές μειωμένης συνέχειας: Έστω ο δεσμός A με συντεταγμένες (3.5,5.5) στο χώρο δεικτών. (α) Ο δεσμός έχει φορέα (μη μηδενικές συναρτήσεις ανάμειξης) εντός τις πράσινης περιοχής $[\xi_1^2, \xi_1^5] \times [\xi_2^3, \xi_2^7]$. Σχεδιάζοντας όλες τις τιμές που περιέχονται στα τοπικά κομβοδιανύσματα $\Xi_1^A = \{\xi_1^2, \xi_1^3, \xi_1^4, \xi_1^5\}$ και $\Xi_2^A = \{\xi_2^3, \xi_2^5, \xi_2^6, \xi_2^7\}$ δημιουργείται το πλέγμα των διακεκομμένων κόκκινων γραμμών. (β) Αν κάποια από αυτές τις γραμμές δεν είναι ήδη ακμή στο (α), τότε προστίθεται στο πλέγμα T-spline. [14]

Στα T-spline, τα στοιχεία ορίζονται ως η ένωση όλων των ακμών και των γραμμών μειωμένης συνέχειας που σχηματίζουν περιοχές με μη μηδενικό εμβαδόν στο παραμετρικό χωρίο ξ_l^u (Σχήμα 3-6). Το T-mesh που συμπεριλαμβάνει τις γραμμές μειωμένης συνέχειας ονομάζεται *στοιχειώδες T-mesh* (elemental T-mesh), καθώς σε αυτό ορίζονται τα *στοιχεία του T-spline* (T-spline elements). Στα στοιχεία του T-spline οι συναρτήσεις T-spline είναι ομαλές (C^∞ -συνέχεια) και είναι εκείνες οι ορθογώνιες υπο-περιοχές, στις οποίες μπορεί να πραγματοποιηθεί η αριθμητική ολοκλήρωση Gauss. Επισημαίνεται ότι στην περίπτωση ύπαρξης διασταύρωσης T, τα στοιχεία T-spline δεν ταυτίζονται με τα στοιχεία του T-mesh, δηλαδή με τις ορθογώνιες περιοχές που ορίζονται αποκλειστικά από τις ακμές του πλέγματος T-spline.



Σχήμα 3-7 Καθορισμό των τοπικών κομβοδιανυσμάτων για ένα πλέγμα T-spline (α) άρτιου ($p = 2$) και (β) περιττού ($p = 3$) βαθμού: κάθε φορά που μια ακμή διασταυρώνεται και στις 4 κατευθύνσεις, η αντίστοιχη παραμετρική τιμή προστίθεται στο τοπικό κομβοδιάνυσμα. (α) Τα τοπικά κομβοδιανύσματα για το μπλε δεσμό A είναι $\Xi_1^A = \{\xi_1^2, \xi_1^3, \xi_1^4, \xi_1^5\} = \left\{0, \frac{1}{2}, 1, 1\right\}$ και $\Xi_2^A = \{\xi_2^3, \xi_2^5, \xi_2^6, \xi_2^7\} = \left\{\frac{1}{3}, \frac{2}{3}, 1, 1\right\}$. (β) Τα τοπικά

κομβοδιανύσματα για το μπλέ δεσμό B είναι $\Xi_1^B = \{\xi_1^1, \xi_1^1, \xi_1^2, \xi_1^4, \xi_1^5\} = \{0,0,0,1,1\}$ και $\Xi_1^B = \{\xi_2^1, \xi_2^1, \xi_2^2, \xi_2^3, \xi_2^5\} = \left\{0,0,0,\frac{1}{3},\frac{2}{3}\right\}$. [14]

4 ΚΑΤΆΛΛΗΛΑ-ΠΡΟΣ-ΑΝΆΛΥΣΗ T-SPLINE

Τα κατάλληλα-προς-ανάλυση T-spline (analysis-suitable T-spline) αποτελούν ένα χρήσιμο υποσύνολο των T-spline, για το οποίο γνωρίζουμε με βεβαιότητα ότι οι συναρτήσεις ανάμειξης είναι γραμμικά ανεξάρτητες. Ο όρος εισήχθη για πρώτη φορά το 2010 από τους Li κ.ά. στο [15]. Η θεωρία των κατάλληλων προς ανάλυση T-spline που παρουσιάζεται παρακάτω έχει εξαχθεί από [11, 9, 15]. Οι κατάλληλες-προς-ανάλυση συναρτήσεις T-spline διατηρούν σημαντικές μαθηματικές ιδιότητες των NURBS, ενώ παρέχουν τη δυνατότητα αποδοτικότερων τοπικών εκλεπτύνσεων, κατά τις οποίες δεν προστίθενται στο πλέγμα πολλά περισσότερα περιττά σημεία ελέγχου [12].

Είναι ήδη γνωστό ότι όλα τα T-spline διαθέτουν τις παρακάτω ιδιότητες:

- Η βάση των ρητών συναρτήσεων ικανοποιεί τη διαμέριση της μονάδας.
- Κάθε συνάρτηση είναι μη αρνητική.
- Ικανοποιείται η ιδιότητα της κυρτής θήκης.
- Είναι εφικτή η τοπική εκλέπτυνση.

Αν και για τα περισσότερα T-spline ικανοποιείται, επιπλέον, η γραμμική ανεξαρτησία των συναρτήσεων ανάμειξης T-spline, έχει αποδειχθεί ότι υπάρχουν T-spline στα οποία εμφανίζεται γραμμική εξάρτηση μεταξύ των συναρτήσεων ανάμειξης [16]. Υπάρχει, όμως, ένα υποσύνολο T-spline, τα επονομαζόμενα *κατάλληλα-προς-ανάλυση* (analysis suitable) T-spline, στα οποία οι συναρτήσεις ανάμειξης T-spline είναι πάντα γραμμικά ανεξάρτητες, ανεξάρτητα από την επιλογή των κομβοδιαστημάτων του πλέγματος [50], [12]. Δηλαδή, ο χαρακτηρισμός ενός T-spline ως «κατάλληλο-προς-ανάλυση» καθορίζεται αποκλειστικά και μόνο από την τοπολογία του πλέγματος T-spline. Οι χώροι των κατάλληλων-προς-ανάλυση T-spline ορίζονται πάνω σε ένα περιορισμένο σύνολο επιτρεπτών τοπολογιών για το T-mesh, οι οποίες περιγράφονται με τη βοήθεια των *επεκτάσεων* των διασταυρώσεων T στα T-mesh. Σημειώνεται, δε, ότι σε πλέγματα T-spline με συγκεκριμένη τοπολογία, οι πολυωνυμικές συναρτήσεις βάσης ενός κατάλληλου-προς-ανάλυση T-spline ικανοποιούν και την ιδιότητα της διαμέρισης της μονάδας, Τα πλέγματα αυτά καλούνται *αποδεκτά* πλέγματα T-spline (admissible T-mesh) και θα συζητηθούν στο Κεφάλαιο 4.4.

Επίσης, αξίζει να τονιστεί ότι ο όρος «βάση» στην περίπτωση ενός T-spline με γραμμικά εξαρτημένες συναρτήσεις T-spline χρησιμοποιείται καταχρηστικά. Οι γραμμικά εξαρτημένες συναρτήσεις T-spline δεν συνιστούν βάση, ούτε μπορούν να χρησιμοποιηθούν σε μεθόδους ισογεωμετρικής ανάλυσης, καθώς οδηγούν στο σχηματισμό μη αντιστρέψιμων πινάκων. [16] [15] Τα T-spline που μπορούν να αξιοποιηθούν στην ανάλυση περιλαμβάνουν τα κατάλληλα-προς-ανάλυση T-spline, αλλά δεν περιορίζονται μόνο σε αυτά. Η ταξινόμηση τους μπορεί να πραγματοποιηθεί με τη βοήθεια του τελεστή εξαγωγής Bézier, όπως θα παρουσιαστεί στο Κεφάλαιο 7.

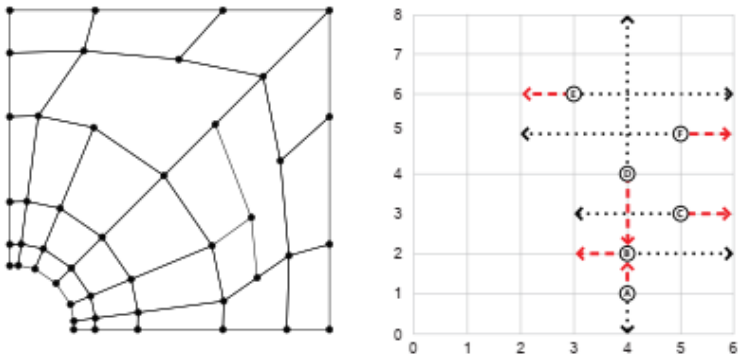
4.1 Επεκτάσεις των διασταυρώσεων T

Η *επέκταση* μιας διασταύρωσης T αποτελείται από μια *επέκταση έδρας* (face extension) και μια *επέκταση ακμής* (edge extension– όταν αυτή υπάρχει). Ορίζουμε τις επεκτάσεις έδρας και ακμής ως κλειστά, προσανατολισμένα ευθύγραμμα τμήματα με σημείο εφαρμογής τη διασταύρωση T. Το

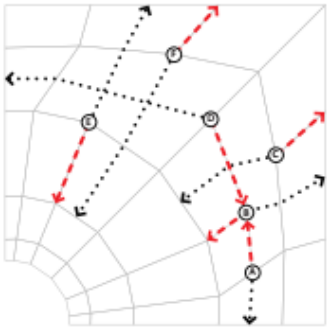
επεκτεταμένο T-mesh σχηματίζεται με την προσθήκη όλων των επεκτάσεων των διασταυρώσεων T στο T-mesh.

Η *επέκταση έδρας* δημιουργείται με τον σχηματισμό ενός προεκτεινόμενου διανύσματος στην κατεύθυνση της ακμής που λείπει από τη διασταύρωση μέχρι την τομή δυο κάθετων ακμών ή κορυφών του πλέγματος. Η *επέκταση ακμής* δημιουργείται έπειτα μόνο αν υπάρχει ακμή στη διασταύρωση με κατεύθυνση αντίθετη από εκείνη της επέκτασης έδρας. Σε αυτή την περίπτωση, η επέκταση θα οριστεί ως ένα κατευθυνόμενο διάνυσμα με πέρας την αντίθετη κορυφή της ακμής. Καθώς, οι επεκτάσεις των διασταυρώσεων T είναι κλειστά ευθύγραμμα τμήματα, είναι πιθανό οι οριζόντιες και κάθετες επεκτάσεις είτε να τέμνονται είτε το ένα άκρο της μιας επέκτασης να ακουμπάει σε ένα άκρο ή και στα δύο άκρα της άλλης επέκτασης.

(α) (β)



(α) (β)

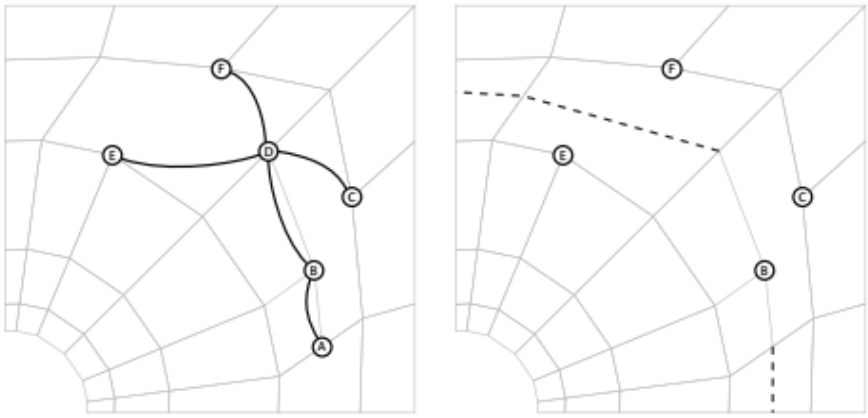


(γ)

Σχήμα 4-1 Το επεκτεταμένο T-mesh, T_{ext}, στο χώρο δεικτών και στο φυσικό χωρίο. Τα διάστικτα βέλη (μαύρο χρώμα) αντιπροσωπεύουν τις επεκτάσεις έδρας, ενώ τα διακεκομμένα βέλη (κόκκινο χρώμα) συμβολίζουν τις επεκτάσεις ακμής. Οι διασταυρώσεις T συμβολίζονται με μεγάλους κύκλους. (α) Ένα T-mesh T με έξι διασταυρώσεις T. (β) Το επεκτεταμένο T-mesh που σχηματίζεται από το T και τις επεκτάσεις των διασταυρώσεων Τα στο χώρο δεικτών. (γ) Το επεκτεταμένο T-mesh στο φυσικό χώρο. Τονίζεται ότι η κατεύθυνση κάθε επέκτασης καθορίζεται στο χώρο δεικτών του T-mesh. [11]

4.2 Το γράφημα επεκτάσεων

Οι τομές που σχηματίζουν οι επεκτάσεις των διασταυρώσεων T σε ένα επεκτεταμένο πλέγμα T-spline T_{ext} μπορούν να απεικονιστούν σε ένα μη κατευθυνόμενο γράφημα. Το γράφημα αυτό ονομάζεται *γράφημα επεκτάσεων* και συμβολίζεται με $E(T_{ext})$. Κάθε κόμβος (node) στο E αντιστοιχεί σε μια επέκταση διασταύρωσης T στο πλέγμα T_{ext}. Αν δύο επεκτάσεις του T_{ext} τέμνονται, τότε σχεδιάζεται μια ακμή στο $E(T_{ext})$ μεταξύ των αντίστοιχων κόμβων. Το γράφημα επεκτάσεων για το επεκτεταμένο T-mesh του Σχήμα 4-1$^{\beta}$ φαίνεται στο Σχήμα 4-2$^{\alpha}$. Σε αυτή την περίπτωση, υπάρχουν πέντε τομές, οι οποίες αναπαρίστανται στο γράφημα με πέντε ακμές.



(α) (β)

Σχήμα 4-2 Το γράφημα επεκτάσεων, $E(T_{ext})$.(α) Το γράφημα επεκτάσεων για το T_{ext} του Σχήμα 4-1(β) και 4-1(γ). Οι πέντε ακμές στο γράφημα αντιστοιχούν στις πέντε τομές μεταξύ των διασταυρώσεων T στο T_{ext}. (β) Το T-mesh στο Σχήμα 4-1(α) μπορεί να μετατραπεί σε «κατάλληλο-προς-ανάλυση» πλέγμα T-spline με την προσθήκη των έντονα διακεκομμένων γραμμών. Το γράφημα επεκτάσεων του νέου T-mesh θα είναι κενό, δηλαδή δε θα έχει ακμές. [11]

4.3 Ορισμός κατάλληλου προς ανάλυση T-spline

Ένα πλέγμα T-spline βαθμού (d_1, d_2) καλείται *κατάλληλο προς ανάλυση* (analysis-suitable, AS T-mesh), αν οι επεκτάσεις των τομών T της μορφής ⊢ και ⊣ δεν τέμνουν τις επεκτάσεις των τομών της μορφής ⊤, ⊥. Ένα T-spline που ορίζεται σε ένα κατάλληλο προς ανάλυση T-mesh καλείται *κατάλληλο προς ανάλυση T-spline* (analysis-suitable T-spline), εν συντομία AS T-spline. [9]

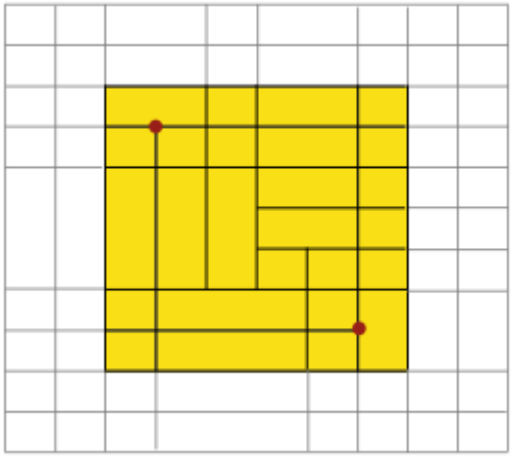
Με άλλα λόγια, το γράφημα επεκτάσεων, $E(T_{ext})$, ενός κατάλληλου-προς-ανάλυση T-mesh είναι ένα κενό γράφημα (δηλαδή, χωρίς ακμές). Το T-mesh στο Σχήμα 4-1$^{\alpha}$ δεν είναι κατάλληλο-προς-ανάλυση, όπως εύκολα διαπιστώνεται από τη μελέτη του γραφήματος επεκτάσεων, στο οποίο υπάρχουν πέντε ακμές Με την προσθήκη των διακεκομμένων ακμών στο πλέγμα, το T-mesh μετατρέπεται σε κατάλληλο-προς-ανάλυση πλέγμα, καθώς το αντίστοιχο γράφημα επεκτάσεων είναι κενό.

Στο [17] γενικεύεται η έννοια του κατάλληλου προς ανάλυση T-spline ώστε να περιλαμβάνει T-spline αυθαίρετων διαστάσεων και βαθμών.

4.4 Αποδεκτό T-mesh

Ένα T-mesh για T-spline βαθμού (d_1, d_2) χαρακτηρίζεται ως *αποδεκτό* (admissible), αν η κορυφή (i, j) στο T-mesh δεν είναι διασταύρωση T της μορφής ⊥ ή ⊤, για $0 \leq i \leq d_1$ ή $c \leq i \leq c + d_1$ και δεν είναι ⊢ ή ⊣, για $0 \leq j \leq d_2$ ή $r \leq j \leq r + d_2$ [9].

Στο Σχήμα 4-3 απεικονίζεται ένα T-mesh τρίτου βαθμού $(d_1 = 3, d_2 = 3)$. Ο χώρος δεικτών του T-spline είναι η ορθογώνια περιοχή $[0,10] \times [0,11]$, επομένως, έχουμε $c = 7$ και $r = 8$. Παρατηρούμε ότι για τις συντεταγμένες $(i, j) = (3,7)$ της διασταύρωσης T της μορφής ⊤, ισχύει $0 \leq i = 3 \leq d_1$, ενώ για τις συντεταγμένες $(i, j) = (7,3)$ της διασταύρωσης T της μορφής ⊣ ισχύει $0 \leq j = 3 \leq d_2$, συνεπώς το T-mesh που απεικονίζεται στο Σχήμα 4-3 είναι μη αποδεκτό.



Σχήμα 4-3 Μη αποδεκτό T-mesh [9]

4.5 Γραμμική ανεξαρτησία και διαμέριση της μονάδας

Αν και έχει αποδειχθεί ότι υπάρχουν T-spline στα οποία οι συναρτήσεις ανάμειξης εμφανίζουν γραμμική εξάρτηση [16], οι Li κ.ά. [15] απέδειξαν ότι τα κατάλληλα-προς-ανάλυση T-spline τρίτου βαθμού διαθέτουν γραμμικά ανεξάρτητες συναρτήσεις T-spline, ενώ οι Zhang κ.ά. [9] μελέτησαν τη γραμμική ανεξαρτησία των συναρτήσεων ανάμειξης για T-spline αυθαίρετου βαθμού (d_1, d_2). Επιπλέον, στην ίδια εργασία αποδείχθηκε ότι η (πολυωνυμική) βάση ενός κατάλληλου-προς-ανάλυση T-spline αυθαίρετου βαθμού ικανοποιεί τη διαμέριση της μονάδας αν το T-mesh είναι αποδεκτό. Επισημαίνεται, επίσης, ότι η ιδιότητα της βάσης (πολυωνυμικής ή ρητής) να διαμερίζει τη μονάδα συνεπάγεται την ικανοποίηση της ιδιότητας της αφινικής συνδιακύμανσης κι ως εκ τούτου την a priori ικανοποίηση των patch test [12]. Η απόδειξη της γραμμικής ανεξαρτησίας βασίστηκε στους πίνακες μετασχηματισμού ενός πλέγματος T-spline σε NURBS (T-spline-to-NURBS transform matrix) και στα γραφήματα επιρροής (influence graph) [15, 9]. Ένας εναλλακτικός και αμεσότερος τρόπος να αποφανθούμε αν ικανοποιούνται οι ιδιότητες της διαμέρισης της μονάδας και της γραμμικής ανεξαρτησίας σε ένα πλέγμα T-spline αυθαίρετου βαθμού προσφέρεται αξιοποιώντας τον τελεστή εξαγωγής Bézier. Η παραπάνω μεθοδολογία προτάθηκε από τους May κ.ά. [14] και παρουσιάζεται στο Κεφάλαιο 7.

5 ΕΙΣΑΓΩΓΗ ΚΌΜΒΟΥ

Η εκλέπτυνση ενός πλέγματος NURBS/B-spline μπορεί να πραγματοποιηθεί με την εισαγωγή κόμβου, την ανύψωση του πολυωνυμικού βαθμού ή συνδυαστικά με τη χρήση και των δύο τεχνικών. Η εισαγωγή κόμβου στο ανά κατεύθυνση κομβοδιάνυσμα αποτελεί ανάλογο της εκλέπτυνσης h της Ανάλυσης Πεπερασμένων Στοιχείων, η ανύψωση του πολυωνυμικού βαθμού αντιστοιχεί στην εκλέπτυνση p. Μερικές φορές εφαρμόζουμε και τις δυο τεχνικές, πραγματοποιώντας, δηλαδή την επονομαζόμενη εκλέπτυνση k, η οποία δεν αντιστοιχίζεται σε μια διαδικασία της Ανάλυσης Πεπερασμένων Στοιχείων. Από τις παραπάνω μεθόδους, στην παρούσα εργασία θα παρουσιαστεί η εισαγωγή κόμβου. Παρακάτω θα μελετηθεί η περίπτωση εισαγωγής κόμβου στην περίπτωση μιας καμπύλης B-spline και έπειτα θα επεκταθεί στην πολυδιάστατη περίπτωση. Η διαδικασία εισαγωγής κόμβου στα NURBS είναι πανομοιότυπη με εκείνη των B-spline, που περιγράφεται παρακάτω με μόνη διαφορά ότι η εισαγωγή των κόμβων πραγματοποιείται στα προβολικά σημεία ελέγχου (σημεία ελέγχου της προβολικής καμπύλης B-spline) και όχι στα σημεία ελέγχου των NURBS αυτά καθαυτά.

Κατά την υλοποίηση της εκλέπτυνσης, οι κόμβοι που προστίθενται στο κομβοδιάνυσμα οφείλουν να αφήνουν αναλλοίωτες τις γεωμετρικές και παραμετρικές ιδιότητες της καμπύλης. Έστω, λοιπόν, $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$ ένα δοσμένο κομβοδιάνυσμα. Η εισαγωγή ενός νέου κόμβου $\bar{\xi} \in [\xi_k, \xi_{k+1})$ με $k > p$ στο κομβοδιάνυσμα Ξ απαιτεί οι $n+1$ νέες συναρτήσεις βάσης να ορίζονται από τις εξισώσεις (2.1) και (2.2) με βάση το νέο κομβοδιάνυσμα $\Xi = \{\xi_1, \xi_2, \dots, \xi_k, \bar{\xi}, \xi_k, \xi_{k+1}, \xi_{n+p+1}\}$. Τα $m = n+1$ νέα σημεία ελέγχου, $\{\bar{\mathbf{P}}_A\}_{A=1}^{m}$ σχηματίζονται από τα αρχικά σημεία ελέγχου ως

$$\bar{\mathbf{P}}_A = \begin{cases} \mathbf{P}_1, & A = 1, \\ a_A \mathbf{P}_A + (1 - a_A)\mathbf{P}_{A-1}, & 1 < A < m, \\ \mathbf{P}_n, & A = m \end{cases} \tag{5.1}$$

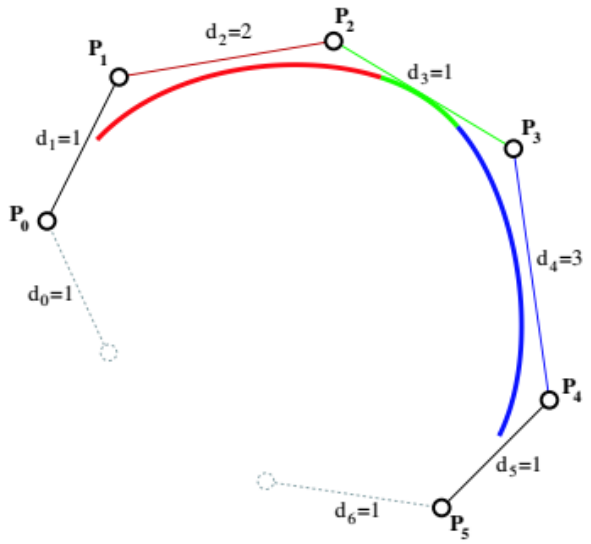
όπου

$$a_A = \begin{cases} 1, & 1 \le A \le k - p, \\ \dfrac{\bar{\xi} - \xi_A}{\xi_{A+p} - \xi_A}, & k - p + 1 \le A \le k, \\ 0, & A \ge k + 1. \end{cases} \tag{5.2}$$

Οι τιμές των κόμβων που προστίθενται μπορεί να έχουν πολλαπλότητα μεγαλύτερη της μονάδας, αν και η αυξημένη πολλαπλότητα συνεπάγεται την μείωση της τάξης συνέχειας της βάσης κατά μια μονάδα για κάθε επανάληψη της κομβικής τιμής. Παρ' όλα αυτά, επιλέγοντας σημεία ελέγχου σύμφωνα με τις εξισώσεις (5.1) και (5.2), η συνέχεια της καμπύλης παραμένει σταθερή, καθώς η τελική καμπύλη που σχηματίζεται από αυτά είναι η ίδια με την αρχική. Στην περίπτωση καμπύλης NURBS, ο υπολογισμός των νέων προβολικών σημείων ελέγχου της καμπύλης B-spline, από την οποία κατασκευάζεται η καμπύλη NURBS, υλοποιείται με βάση τις εξισώσεις (5.1) και (5.2) κι έπειτα τα νέα σημεία ελέγχου της καμπύλης NURBS προκύπτουν με την διαίρεση των συντεταγμένων κάθε νέου σημείου ελέγχου με τον αντίστοιχο συντελεστή βάρους του.

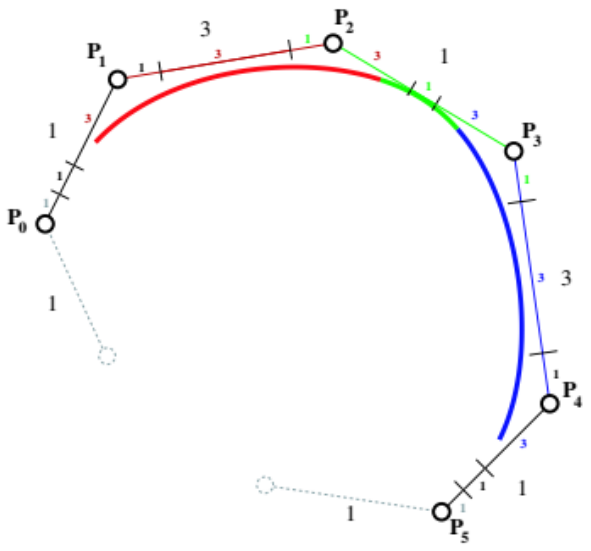
5.1 Διαχωρισμός κομβοδιαστημάτων

Παρακάτω παρουσιάζεται μια εναλλακτική προσέγγιση της εισαγωγικής κόμβου με τη χρήση κομβοδιαστημάτων στη θέση των κομβικών τιμών, όπως διατυπώθηκε στο [8]. Καθώς, η εισαγωγή μιας κομβικής τιμής ανάμεσα σε δυο κόμβους ισοδυναμεί με το διαχωρισμό σε δυο τμήματα του κομβοδιαστήματος που εκτείνεται ανάμεσα τους, η εισαγωγή κόμβων υπό αυτό τον πρίσμα μπορεί να ιδωθεί ως «διαχωρισμός κομβοδιαστημάτων» (interval splitting).



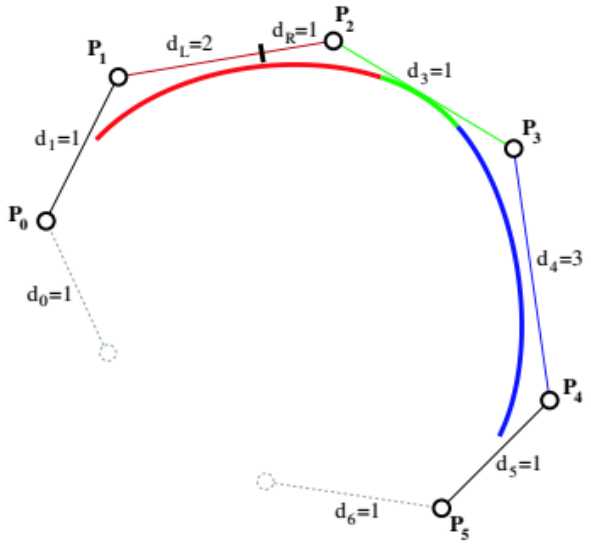
Σχήμα 5-1 Μια καμπύλη NURBS πριν την εισαγωγή του κόμβου με σημειωμένα τα κομβικά διαστήματα d_i. [8]

Η παραπάνω διαδικασία υλοποιείται στην καμπύλη τρίτου βαθμού ($p = 3$) (Σχήμα 5-1) για την οποία το κομβοδιάστημα $d_2 = 3$ χωρίζεται σε δύο διαστήματα με μήκος 2 και 1. Αρχικά, σε κάθε ακμή της καμπύλης NURBS σημειώνονται το μήκος του κομβοδιαστήματος της ακμής και το μήκος των κομβοδιαστημάτων των γειτονικών ακμών, όπως στο Σχήμα 5-2. Επισημαίνεται ότι τα κομβοδιαστήματα που σημειώνονται στις ακμές σχεδιάζονται σε αναλογία με το μήκος τους. Επιπλέον, τα διαστήματα που σχετίζονται με μια ακμή εμφανίζονται συνολικά σε τρεις ακμές (στην ακμή με την οποία σχετίζονται και στις δυο γειτονικές της).



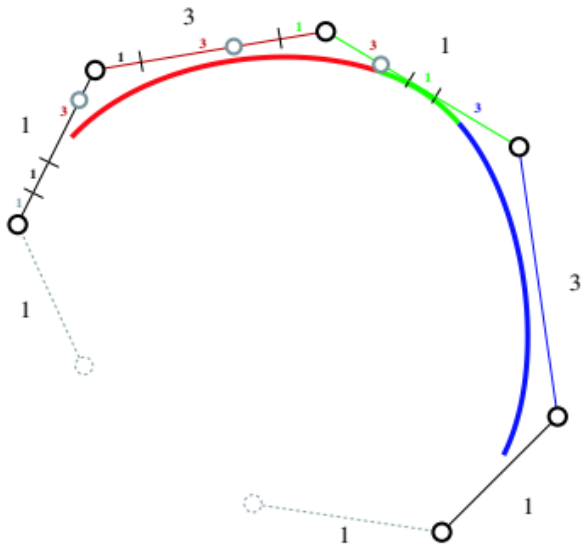
Σχήμα 5-2 Η καμπύλη NURBS με σημειωμένες τις ακμές για την προετοιμασία της εισαγωγής του κόμβου. Καθεμία από τις ακμές του πολυγώνου ελέγχου σημειώνεται σύμφωνα με τα γειτονιά κομβοδιαστήματα. [8]

Στη συνέχεια, επιλέγεται μια παραμετρική απόσταση σε μια από τις ακμές του πολυγώνου ελέγχου, στην οποία θα πραγματοποιηθεί η εισαγωγή κόμβου και η οποία θα καλείται ακμή εισαγωγής, όπως στο Σχήμα 5-3.



Σχήμα 5-3 Η επιλογή της θέσης της παραμέτρου εισαγωγής στην καμπύλης NURBS. Η επιλεγμένη θέση είναι τα δύο τρίτα της απόστασης της κόκκινης ακμής με μήκος διαστήματος d_2 και χωρίζει το διάστημα στα d_L και d_R. [8]

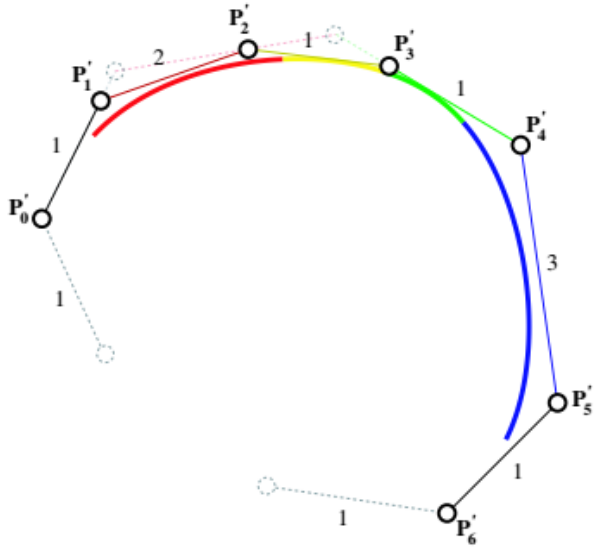
Έπειτα, προσδιορίζεται η παραμετρική θέση στις τρεις σημειωμένες ακμές, συμπεριλαμβανομένου του διαστήματος που σχετίζεται με την εισαγωγή του κόμβου (Σχήμα 5-4).



Σχήμα 5-4 Σήμανση της παραμέτρου εισαγωγής στις σημειωμένες ακμές. Η παράμετρος εισαγωγής είναι σημειωμένη σε κάθε μια από τις σχετιζόμενες ακμές και τα τρία σημεία υποδεικνύουν τις θέσεις των νέων σημείων ελέγχου NURBS μετά την εισαγωγή. [8]

Τέλος, υπολογίζονται οι θέσεις των σημείων ελέγχου στις προσδιορισμένες παραμετρικές θέσεις– τα δύο σημεία ελέγχου τα οποία σχετίζονται με την ακμή, στην οποία προστέθηκε ο νέος κόμβος μετακινούνται στις θέσεις των αντίστοιχων γειτονικών ακμών και το νέο σημείο ελέγχου τοποθετείται στην προσδιορισμένη θέση στην ακμή εισαγωγής, όπως απεικονίζεται στο Σχήμα 5-5. Αυτή η γεωμετρική περιγραφή παρέχει ένα γραφικό μέσο για την εξαγωγή μιας αναλυτικής γραφής της εισαγωγής κόμβου. Από το Σχήμα 5-4, μπορούν να συναχθούν οι γραμμικοί συνδυασμοί των αρχικών σημείων ελέγχου που συνθέτουν τις θέσεις των νέων, χρησιμοποιώντας το συμβολισμό στο Σχήμα 5-3:

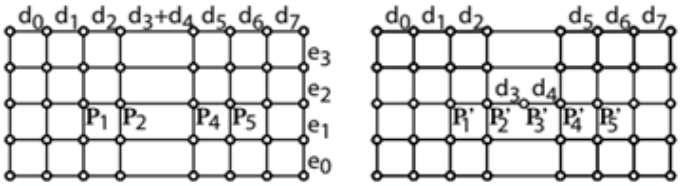
$$
\begin{aligned}
\mathbf{P}_1' &= \frac{d_R}{d_0 + d_1 + d_2}\mathbf{P}_0 + \frac{d_0 + d_1 + d_L}{d_0 + d_1 + d_2}\mathbf{P}_1 \\
\mathbf{P}_k' &= \frac{d_R + d_3}{d_1 + d_2 + d_3}\mathbf{P}_1 + \frac{d_1 + d_L}{d_1 + d_2 + d_3}\mathbf{P}_2 \\
\mathbf{P}_2' &= \frac{d_R + d_3 + d_4}{d_2 + d_3 + d_4}\mathbf{P}_2 + \frac{d_L}{d_2 + d_3 + d_4}\mathbf{P}_3.
\end{aligned}
\tag{5.3}
$$



Σχήμα 5-5 Η τελική καμπύλη NURBS μετά την ολοκλήρωση της εισαγωγής στο επιλεγμένο παραμετρικό σημείο. [8]

Ο εν λόγω αλγόριθμος λειτουργεί αποκλειστικά για εσωτερικούς κόμβους πολλαπλότητας ≥ 1 και δεν μπορεί να αυξήσει την πολλαπλότητα στα άκρα.

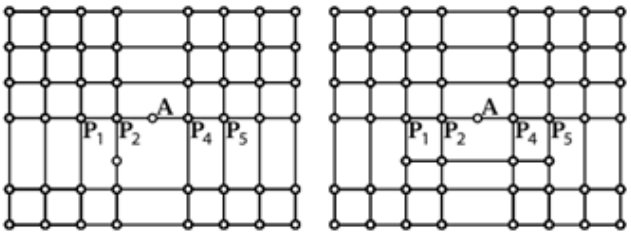
Επιπλέον, στην περίπτωση εισαγωγής κόμβου σε οριζόντια ακμή ενός διδιάστατου πλέγματος T-spline τρίτου βαθμού θα πρέπει τα τέσσερα γειτονικά η-κομβοδιανύσματα να είναι ίσα, δηλαδή $\boldsymbol{\eta}_1 = \boldsymbol{\eta}_2 = \boldsymbol{\eta}_4 = \boldsymbol{\eta}_5$, όπου $\boldsymbol{\eta}_i$ το η-κομβοδιάνυσμα της συνάρτησης B_i (Σχήμα 5-6). Αντίστοιχα, αν η εισαγωγή εφαρμοζόταν σε κάθε ακμή, θα έπρεπε τα τέσσερα γειτονικά ξ-κομβοδιανύσματα να ήταν ίσα. [5]



(α) Πριν την εισαγωγή (β) Μετά την εισαγωγή

Σχήμα 5-6 Εισαγωγή κόμβου σε T-mesh [5]

Επομένως, δεν είναι πάντα εφικτό να εισαγάγουμε απευθείας τον κόμβο σε οποιαδήποτε ακμή του πλέγματος. Για παράδειγμα, στο Σχήμα 5-7$^{\alpha}$, ο παραπάνω κανόνας δεν επιτρέπει να εισαχθεί το σημείο **A** στο πλέγμα, καθώς το κομβοδιάνυσμα $\boldsymbol{\eta}_2$ είναι διαφορετικό από τα $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_4$ και $\boldsymbol{\eta}_5$. Παρ' όλα αυτά, αν εισαχθούν προηγουμένως σημεία ελέγχου κάτω από τα $\mathbf{P}_1$, $\mathbf{P}_4$ και $\mathbf{P}_5$, όπως φαίνεται στο Σχήμα 5-7$^{\beta}$, η προσθήκη του **A** μπορεί να πραγματοποιηθεί χωρίς πρόβλημα. [5]



(α) Το **A** δεν μπορεί να προστεθεί. (β) Το **A** μπορεί να προστεθεί.

Σχήμα 5-7 Εισαγωγή κόμβου σε T-mesh [5]

5.2 Αλγόριθμος του Oslo

Στην περίπτωση διαδοχικής εισαγωγής κομβικών τιμών, έστω m το πλήθος, στο αρχικό κομβοδιάνυσμα Ξ, το τελικό κομβοδιάνυσμα θα είναι $\bar{\Xi} = \{\bar{\xi}_1 = \xi_1, \bar{\xi}_2, \dots, \bar{\xi}_{n+m+p+1} = \xi_{n+p+1}\}$ με $\Xi \subset \bar{\Xi}$. Οι $n + m$ νέες συναρτήσεις βάσης σχηματίζονται με εφαρμογή των αναδρομικών σχέσεων Cox-de Boor (2.1), (2.2), χρησιμοποιώντας το νέο κομβοδιάνυσμα $\bar{\Xi}$. Τα νέα $n + m$ σημεία ελέγχου $\bar{P} = \{\bar{\mathbf{P}}_1, \bar{\mathbf{P}}_2, \dots, \bar{\mathbf{P}}_{n+m}\}^T$ σχηματίζονται από τα αρχικά σημεία ελέγχου $P = \{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n\}^T$ μέσω του ακόλουθου γραμμικού μετασχηματισμού, ο οποίος καλείται αλγόριθμος του Oslo:

$$\bar{P} = \mathbf{T}^p P, \tag{5.4}$$

όπου

$$T_{ij} = \begin{cases} 1, & \text{αν } \bar{\xi}_i \in [\xi_j, \xi_{j+1}), \\ 0, & \text{διαφορετικά} \end{cases} \tag{5.5}$$

και

$$T_{ij}^{q+1} = \frac{\bar{\xi}_{i+q} - \xi_j}{\xi_{j+q} - \xi_j} T_{ij}^q + \frac{\xi_{j+q+1} - \bar{\xi}_{i+q}}{\xi_{j+q+1} - \xi_{j+1}} T_{ij+1}^q$$
$$\text{για } q = 0,1,2, \dots, p-1. \tag{5.6}$$

Αν επιθυμούμε να εισάγουμε κόμβους σε μια επιφάνεια ή σε ένα στερεό αντί σε καμπύλη, ακολουθούμε μια παρόμοια διαδικασία. Ο πίνακας $\mathbf{T}^p$ σχηματίζεται όπως παραπάνω, χρησιμοποιώντας τα κομβοδιανύσματα Ξ^l και $\bar{\Xi}^l$. Τα νέα σημεία ελέγχου υπολογίζονται εφαρμόζοντας την εξίσωση (5.6) σε κάθε γραμμή ή στήλη του πλέγματος ελέγχου. Για παράδειγμα, έστω μια καμπύλη B-spline στην οποία προστίθενται m κόμβοι στην παραμετρική κατεύθυνση $l = 1$. Δηλαδή, το κομβοδιάνυσμα Ξ^1 θα εκλεπτυνθεί, ενώ το κομβοδιάνυσμα Ξ^2 θα παραμείνει ίδιο. Τα νέα σημεία ελέγχου της επιφάνειας υπολογίζονται από τη σχέση

$$\bar{\mathbf{P}}_{ik} = \sum_{j}^{n_1} T_{ij}^p \mathbf{P}_{jk}, \tag{5.7}$$

για $i = 1,2, \dots, n_1 + m$ και $k = 1,2, \dots, n_2$. Οι νέες συναρτήσεις τανυστικού γινόμενου παράγονται κατά το γνωστό τρόπο από τα κομβοδιανύσματα $\bar{\Xi}_1$ και $\bar{\Xi}_2$. [4]

5.3 Υποδιαίρεση συναρτήσεων

Η εισαγωγή κόμβων μπορεί να ιδωθεί και ως υποδιαίρεση των συναρτήσεων (function subdivision). [4] Για τη διερεύνηση αυτής της οπτικής, συμβολίζουμε με $\mathcal{S}$ το χώρο όλων των καμπυλών που μπορούν να κατασκευαστούν από το αρχικό κομβοδιάνυσμα Ξ και με $\bar{\mathcal{S}}$ το χώρο όλων των καμπυλών που μπορούν να κατασκευαστούν από το τελικό κομβοδιάνυσμα $\bar{\Xi}$. Αν η γεωμετρία και η παραμετροποίηση της παραμείνουν σταθερές, θα ισχύει

$$\mathcal{S} \subset \bar{\mathcal{S}}. \tag{5.8}$$

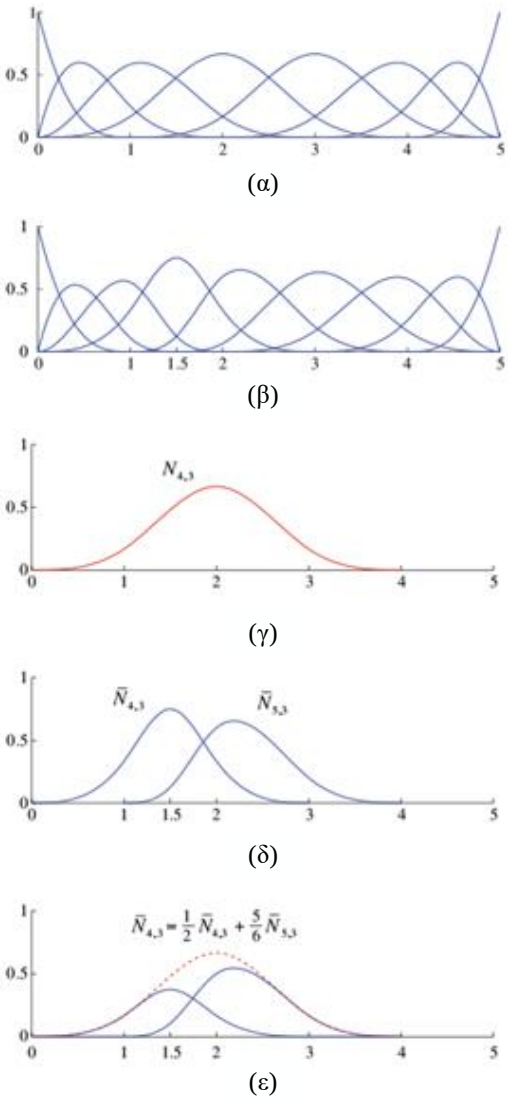
Ένας φυσικός τρόπος να διασφαλιστεί η παραπάνω σχέση είναι να απαιτήσουμε κάθε αρχική συνάρτηση βάσης να μπορεί να εκφραστεί ως γραμμικός συνδυασμός των συναρτήσεων της εκλεπτυσμένης βάσης. Η έννοια της υποδιαίρεσης των συναρτήσεων ενυπάρχει ήδη στην εξίσωση (5.4). Για να γίνει αυτό αντιληπτό, θεωρούμε το διάνυσμα $N = \{N_1(\xi), N_2(\xi), \dots, N_n(\xi)\}^T$, το οποίο περιέχει όλες τις συναρτήσεις βάσης, οι οποίες παράγονται από το Ξ και το $\bar{N} = \{\bar{N}_1(\xi), \bar{N}_2(\xi), \dots, \bar{N}_n(\xi)\}^T$ το οποίο περιέχει όλες τις συναρτήσεις βάσης που παράγονται από το $\bar{\Xi}$, ενώ ο πολυωνυμικός βαθμός των δυο βάσεων είναι κοινός κι ως εκ τούτου δεν αναγράφεται. Επομένως, η καμπύλη B-spline γράφεται ως:

$$\mathbf{C}(\xi) = P^T N = \bar{P}^T \bar{N} = (\mathbf{T}^p P)^T \bar{N} = P^T((\mathbf{T}^p)^T \bar{N}) \tag{5.9}$$

κι ως εκ τούτου,

$$N = (\mathbf{T}^p)^T \bar{N}. \tag{5.10}$$

Ένα παράδειγμα της παραπάνω προσέγγισης παρουσιάζεται στο Σχήμα 5-8. Το αρχικό κομβοδιάνυσμα είναι το $\Xi = \{0,0,0,0,1,2,3,4,5,5,5,5\}$, από το οποίο κατασκευάζεται μια κυβική βάση, η οποία απεικονίζεται στο Σχήμα 5-8. Μετά την εισαγωγή της νέας κομβικής τιμής $\bar{\xi} = 1.5$ στο κομβοδιάνυσμα, κατασκευάζεται μια νέα βάση σύμφωνα με τις σχέσεις (2.1) και (2.2), όπως φαίνεται στο Σχήμα 5-8. Κάθε συνάρτηση στην αρχική βάση, όπως εκείνη στο Σχήμα 5-8$^{\gamma}$, μπορεί να αναπαρασταθεί ως γραμμικός συνδυασμός των συναρτήσεων της εκλεπτυσμένης βάσης. Σε αυτή την περίπτωση, οι δυο συναρτήσεις στο Σχήμα 5-8$^{\delta}$ μπορούν να συνδυαστούν για να παράγουν την αρχική συνάρτηση, όπως φαίνεται στο Σχήμα 5-8$^{\varepsilon}$. [4]



Σχήμα 5-8 Υποδιαίρεση συναρτήσεων. (α) Οι συναρτήσεις B-spline τρίτου βαθμού ($p = 3$) που κατασκευάζονται από το κομβοδιάνυσμα $\Xi = \{0,0,0,0,1,2,3,4,5,5,5,5\}$. (β) Οι νέες συναρτήσεις B-spline που κατασκευάζονται από το $\Xi = \{0,0,0,0,1,1.5,2,3,4,5,5,5,5\}$ και $p = 3$. (γ) Έστω, για παράδειγμα, η συνάρτηση $N_{4,3}$ από την αρχική βάση, η οποία έχει φορέα στο $[0,4]$ και προσεγγίζει την αρχή των αξόνων με μηδενική δεύτερη παράγωγο. (δ) Η νέα βάση έχει δύο συναρτήσεις, $\bar{N}_{4,3}$ και $\bar{N}_{5,3}$, οι οποίες έχουν επίσης φορέα στο $[0,4]$ και προσεγγίζουν την αρχή των αξόνων με μηδενική παράγωγο. (ε) Η αρχική συνάρτηση $N_{4,3}$ μπορεί να εκφραστεί ως γραμμικός συνδυασμός των $\bar{N}_{4,3}$ και $\bar{N}_{5,3}$ με τους συντελεστές $\frac{1}{2}$ και $\frac{5}{6}$ αντίστοιχα. [4]

Η σχέση (5.10) είναι απλώς ένας ισοδύναμος τρόπος να προσεγγιστεί η εισαγωγή κόμβων σε κομβοδιάνυσμα, εστιάζοντας, όμως, στην υποδιαίρεση των συναρτήσεων βάσης αντί στην εκλέπτυνση του κομβοδιανύσματος και την μετέπειτα κατασκευή των συναρτήσεων από το νέο κομβοδιάνυσμα. Ωστόσο, η υποδιαίρεση των συναρτήσεων βάσης ως προσέγγιση θα φανεί ιδιαίτερα χρήσιμη στην εκλέπτυνση των συναρτήσεων NURBS/T-spline και στην οποία θα επανέλθουμε στο Κεφάλαιο 6, όπου θα παρουσιαστεί η Αποσύνθεση Bézier.

6 ΑΠΟΣΥΝΘΕΣΗ BÉZIER ΣΕ NURBS & T-SPLINE

Όπως έγινε σαφές στα Κεφάλαια 2 και 3, ο φορέας των συναρτήσεων NURBS και T-spline δεν περιορίζεται σε ένα στοιχείο, αλλά εκτείνεται σε πολλά στοιχεία και οι ίδιες οι συναρτήσεις ορίζονται σε ολόκληρο το παραμετρικό χωρίο. Αυτό το χαρακτηριστικό των συναρτήσεων NURBS και T-spline αποτελεί μειονέκτημα στην ενσωμάτωση τους σε έναν κώδικα πεπερασμένων στοιχείων, όπου οι συναρτήσεις μορφής θα ορίζονταν τοπικά σε κάθε στοιχείο του πλέγματος. Επιπλέον, για την υλοποίηση της ολοκλήρωσης Gauss σε συναρτήσεις CAD, τα σημεία Gauss του γονικού στοιχείου θα πρέπει να μετασχηματίζονται σε παραμετρικά σημεία Gauss, δηλαδή να απεικονίζονται στο παραμετρικό χωρίο, ώστε να μπορούν υπολογιστούν σε αυτά οι συναρτήσεις NURBS/T-spline. Αυτές οι δυο διαφορές των συναρτήσεων CAD και των συναρτήσεων μορφής FEA δεν επιτρέπουν την ενσωμάτωση τους σε έναν κοινό κώδικα πεπερασμένων στοιχείων, καθώς υπάρχει ένα αλγοριθμικό κενό που αποτρέπει την συνύπαρξη τους. Για τη γεφύρωση αυτού του χάσματος, προτάθηκε η ιδέα της Αποσύνθεσης Bézier από τον Borden [18] για τις συναρτήσεις NURBS και από το Scott [13] για συναρτήσεις T-spline. Σύμφωνα με την Αποσύνθεση Bézier, με την αύξηση της πολλαπλότητας των εσωτερικών κόμβων μέχρι αυτή να γίνει $p+1$, μπορούμε να αποσυνθέσουμε την αρχική βάση NURBS/T-spline σε επιμέρους στοιχεία Bézier. Καθώς η Αποσύνθεση Bézier συνίσταται από διαδοχικές εισαγωγές κόμβων, αποτελεί επί της ουσίας εκλέπτυνση του αρχικού πλέγματος. Επομένως, σύμφωνα με το Κεφάλαιο 5.3, οι αρχικές συναρτήσεις NURBS/T-spline μπορούν να διατυπωθούν ως γραμμικός συνδυασμός των νέων συναρτήσεων Bernstein. Στο παρόν Κεφάλαιο θα παρουσιαστεί η έννοια των ρητών και μη πολυωνύμων Bernstein, τα οποία συνιστούν τη βάση Bézier/Bernstein, καθώς και η έννοια του τελεστή εξαγωγής Bézier για NURBS και T-spline.

6.1 Πολυώνυμα Bernstein

Τα πολυώνυμα Bernstein βαθμού p είναι οι συναρτήσεις που ορίζονται κατά κανόνα στο διάστημα $[0,1]$ (σε κάθε παραμετρικό άξονα, στις πολυδιάστατες περιπτώσεις) με τη χρήση των ανοιχτών κομβοδιανυσμάτων με άκρα 0 και 1, χωρίς εσωτερικούς κόμβους για κάθε διάσταση. Το φυσικό χωρίο το οποίο ορίζεται από τα πολυώνυμα Bernstein ονομάζεται *στοιχείο Bézier* (Bézier element). Στην Ισογεωμετρική Ανάλυση, τα πολυώνυμα Bernstein ορίζονται στο διάστημα $[-1,1]$, ώστε να διευκολύνεται η ολοκλήρωση με τον κανόνα Gauss. Καθώς τα πολυώνυμα Bernstein είναι μια υποπερίπτωση συνάρτησης B-spline βαθμού p, με κομβοδιάνυσμα το διάνυσμα $[-1,\dots,-1,1,\dots,1]$ όπου η πολλαπλότητα των κομβικών τιμών -1 και 1 είναι ίση με $p+1$, μπορούν να οριστούν μέσω της αναδρομικής σχέσης Cox-de Boor ή διαφορετικά στην παρακάτω αναδρομική συμπαγή μορφή:

$$B_{i,p}(\tilde{\xi}) = \frac{1}{2}(1-\tilde{\xi})B_{i,p-1}(\tilde{\xi}) + \frac{1}{2}(1+\tilde{\xi})B_{i-1,p-1}(\tilde{\xi}) \tag{6.1}$$

όπου

$$B_{1,0}(\tilde{\xi}) \equiv 1 \text{ και } B_{1,p}(\tilde{\xi}) \equiv 0 \text{ αν } i < 1 \text{ ή } i > p+1 \tag{6.2}$$

Τα πολυώνυμα Bernstein διατηρούν τις ιδιότητες των συναρτήσεων B-spline, όπως τη μη αρνητικότητα και τη διαμέριση της μονάδας. Επιπλέον, μοιράζονται κοινές ιδιότητες με τα πολυώνυμα Lagrange, όπως η συμμετρία. Στην περίπτωση $p=1$, τα γραμμικά πολυώνυμα

Bernstein ταυτίζονται με τα γραμμικά πολυώνυμα Lagrange, ενώ οποιαδήποτε άλλη περίπτωση ($p > 1$), δεν διατηρείται η σχέση ταύτισης. Εντούτοις, υπάρχει σχέση γραμμικής εξάρτησης μεταξύ των (μη ρητών) πολυωνύμων Bernstein και των πολυωνύμων Lagrange, ενώ οι χώροι που παράγονται από αυτά ταυτίζονται. [3]

Η πρώτη παράγωγος του i-οστού πολυωνύμου Bernstein είναι ίση με

$$B'_{i,p}(\tilde{\xi}) = \frac{dB_{i,p}(\xi)}{d\xi} = \frac{p}{2}\left(B_{i-1,p-1}(\tilde{\xi}) - B_{i,p-1}(\tilde{\xi})\right) \tag{6.3}$$

με $B_{0,p-1}(\tilde{\xi}) \equiv B_{p,p-1}(\tilde{\xi}) = 0$.

Στη μονοδιάστατη περίπτωση, η καμπύλη Bézier σχηματίζεται ως ο γραμμικός συνδυασμός των πολυωνύμων Bernstein και των σημείων ελέγχου, δηλαδή

$$\mathbf{C}(\xi) = \sum_{i=1}^{p+1} B_{i,p}(\tilde{\xi})\mathbf{P}_i = \mathbf{P}^T\mathbf{B}(\tilde{\xi}) \tag{6.4}$$

6.2 Ρητά πολυώνυμα Bernstein

Για τη ακριβή αναπαράσταση της πλειονότητας των κωνικών τομών, υιοθετείται η παρεμβολή με ρητά πολυώνυμα Bernstein. Συγκεκριμένα, μια ρητή καμπύλη Bézier εκφράζεται ως εξής:

$$\mathbf{C}(\tilde{\xi}) = \sum_{i=1}^{p+1} R_{i,p}(\xi)\mathbf{P}_i \tag{6.5}$$

όπου

$$R_{i,p}(\tilde{\xi}) = \frac{B_{i,p}(\tilde{\xi})w_i}{\sum_{i=1}^{p+1} B_{i,p}(\tilde{\xi})w_j} \tag{6.6}$$

είναι το ρητό πολυώνυμο Bernstein ως προς τη μεταβλητή $\tilde{\xi}$, ενώ w_i είναι οι κατάλληλοι συντελεστές βάρους που αντιστοιχούν στα σημεία ελέγχου $\mathbf{P}_i$, για $i = 1, \dots, p+1$.

Αντίστοιχα, η πρώτη παράγωγος του i-οστού ρητού πολυωνύμου Bernstein είναι ίση με

$$\begin{aligned} R'_{i,p}(\tilde{\xi}) &= \frac{dR_{i,p}(\tilde{\xi})}{d\xi} = \frac{B'_{i,p}(\tilde{\xi})w_i \cdot \sum_{j=1}^{p+1} B_{j,p}(\tilde{\xi})w_j - B_{i,p}(\tilde{\xi})w_i \cdot \sum_{j=1}^{p+1} B'_{j,p}(\tilde{\xi})w_j}{\left(\sum_{j=1}^{p+1} B_{j,p}(\tilde{\xi})w_j\right)^2} \\ &= \frac{B'_{i,p}(\tilde{\xi})w_i \cdot W(\tilde{\xi}) - R_{i,p}(\tilde{\xi})\, W(\tilde{\xi}) \cdot W'(\tilde{\xi})}{\left(W(\tilde{\xi})\right)^2} = \frac{B'_{i,p}(\tilde{\xi})w_i - R_{i,p}(\tilde{\xi})\, W'(\xi)}{W(\tilde{\xi})} \end{aligned} \tag{6.7}$$

με

$$W(\tilde{\xi}) = \sum_{j=1}^{p+1} B_{j,p}(\tilde{\xi}) w_j \tag{6.8}$$

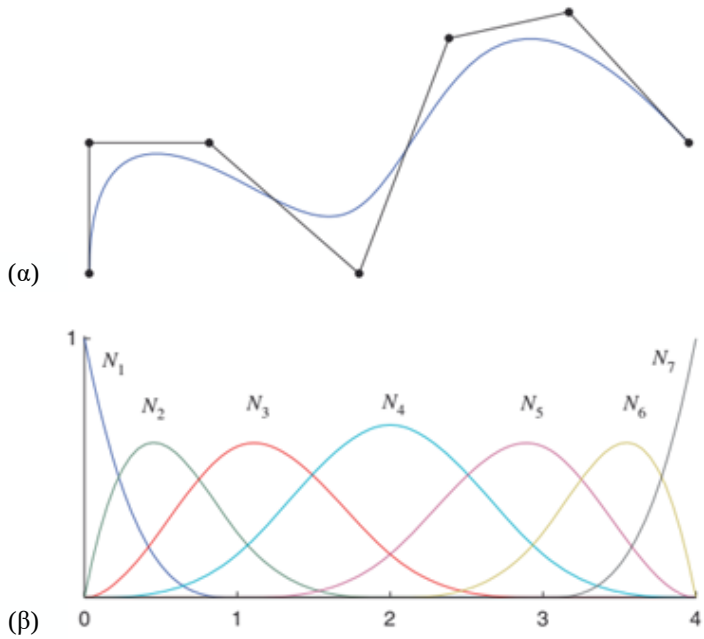
και

$$W'(\tilde{\xi}) = \sum_{j=1}^{p+1} B'_{j,p}(\tilde{\xi}) w_j \tag{6.9}$$

$B_{0,p-1}(\tilde{\xi}) \equiv B_{p,p-1}(\tilde{\xi}) = 0.$

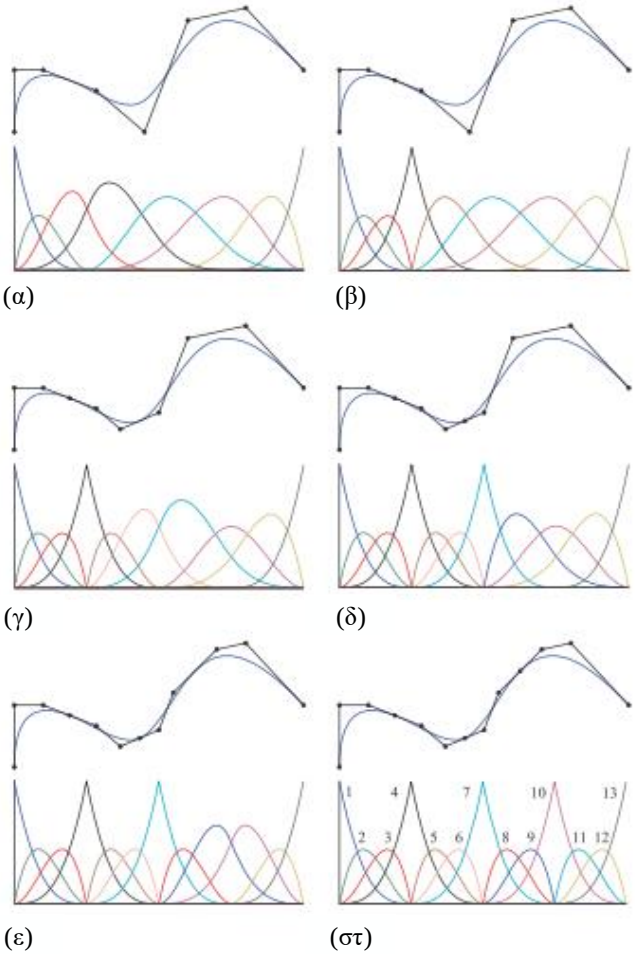
6.3 Αποσύνθεση Bézier και Εξαγωγή Τελεστή Bézier σε βάση NURBS

Για τον υπολογισμό των στοιχείων Bézier μιας βάσης NURBS, χρησιμοποιείται η Αποσύνθεση Bézier. Πρακτικά, η Αποσύνθεση Bézier συνίσταται από την επανάληψη όλων των εσωτερικών κόμβων ενός κομβοδιανύσματος έως ότου κάθε κομβική τιμή τους να αποκτήσει πολλαπλότητα $p + 1$. Ωστόσο, στο συγκεκριμένο πλαίσιο, η πολλαπλότητα p κρίνεται επαρκής για την αναπαράσταση των πολυωνύμων Bernstein, τα οποία αναφέρονται, επίσης, ως συναρτήσεις βάσης Bézier. Παρακάτω ακολουθεί η διερεύνηση ενός παραδείγματος Αποσύνθεσης Bézier για μια μονοδιάστατη καμπύλη B-spline όπως παρουσιάστηκε στο [18]. Στο Σχήμα 6-1$^{\alpha}$ απεικονίζεται μια καμπύλη B-spline τρίτου βαθμού, της οποίας το κομβοδιάνυσμα είναι $\Xi = [0,0,0,0,1,2,3,4,4,4,4]$. Για την αποσύνθεση της καμπύλης στα επιμέρους στοιχεία Bézier της, θα πραγματοποιηθεί εισαγωγή κόμβων σε όλους τους εσωτερικούς κόμβους του κομβοδιανύσματος Ξ, ξεκινώντας από τα αριστερά, μέχρι η πολλαπλότητα των κομβικών τους τιμών να γίνει ίση με p, όπου p ο βαθμός της καμπύλης. Επομένως, πραγματοποιείται εκλέπτυνση με την προσθήκη των κόμβων $\{1,1,2,2,3,3\}$ στο κομβοδιάνυσμα Ξ. Γενικά, σημειώνεται ότι η εισάγοντας τους πραγματοποιείται από τα αριστερά προς τα δεξιά, μειώνεται το συνολικό υπολογιστικό κόστος καθώς ενημερώνονται οι επικαλυπτόμενοι συντελεστές μεταξύ γειτονικών στοιχείων.



Σχήμα 6-1 Κυβική καμπύλη NURBS: (α) η καμπύλη και τα σημεία ελέγχου της και (β) οι συναρτήσεις βάσης της καμπύλης. Το κομβοδιάνυσμα για την καμπύλη είναι το {0,0,0,0,1,2,3,4,4,4,4}. [18]

Στο Σχήμα 6-2 παρουσιάζεται η ακολουθία των συναρτήσεων βάσης και των σημείων ελέγχου που σχηματίζονται με την προσθήκη των νέων κόμβων στο κομβοδιάνυσμα. Κάθε εισαχθείς κόμβος μειώνει την τάξη συνέχειας της βάσης κατά μια μονάδα στη συντεταγμένη του κόμβου. Το αποτέλεσμα των επανειλημμένων προσθηκών είναι η βάση να αποσυντίθεται σε ένα σύνολο C^0 στοιχείων Bézier, στο οποίο κάθε στοιχείο αντιστοιχίζεται σε ένα κομβοδιάστημα του αρχικού κομβοδιανύσματος. Τα σημεία ελέγχου των στοιχείων Bézier υπολογίζονται από τις εξισώσεις (5.1) και (5.2), κάθε φορά που προστίθεται ένας νέος κόμβος. Η επιλογή των συγκεκριμένων σημείων ελέγχου εξασφαλίζει ότι αφ' ενός ότι η γεωμετρία της καμπύλης παραμένει αναλλοίωτη και αφ' ετέρου ότι δε μεταβλήθηκε η τάξη συνέχειας της καμπύλης.



Σχήμα 6-2 Η ακολουθία των συναρτήσεων βάσης που κατασκευάζονται με την προσθήκη των κόμβων {1,1,2,2,3,3} στο κομβοδιάνυσμα για την καμπύλη του Σχήμα 6-1. Το τελικό σύνολο των συναρτήσεων βάσης στο (στ) είναι μια συλλογή τμηματικών κυβικών συναρτήσεων βάσης Bézier. Οι αριθμοί στο (στ) υποδηλώνουν το σύστημα αρίθμησης των συναρτήσεων βάσης Bézier. [18]

Για την αναπαράσταση της καμπύλης NURBS ως προς τα επιμέρους μπεζεριανά στοιχεία της, είναι απαραίτητος ο υπολογισμός του τελεστή εξαγωγής Bézier.

Έστω, λοιπόν, $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$ κομβοδιάνυσμα και $\mathbf{P} = \{\mathbf{P}_A\}_{A=1}^{n}$ ένα σύνολο σημείων ελέγχου τα οποία ορίζουν μια καμπύλη B-spline. Έστω, επίσης, $\{\bar{\xi}_1, \bar{\xi}_2, \dots, \bar{\xi}_m\}$ το σύνολο κόμβων τα οποία πρέπει να προστεθούν για την υλοποίηση της Αποσύνθεσης Bézier για την καμπύλη B-spline. Τότε για κάθε νέο κόμβο $\bar{\xi}_j, j = 1, \dots, m$, ορίζονται τα $a_i^j, i = 1,2, \dots, n + j$ ως ο i-οστός συντελεστής που αντιστοιχεί στον j-εισαγόμενο κόμβο, σύμφωνα με τις εξισώσεις της εισαγωγής κόμβων του Κεφαλαίου 5. Ορίζοντας $\mathbf{C}^j \in \mathbb{R}^{(n+j-1)\times(n+j)}$ ως

$$\mathbf{C}^j = \begin{bmatrix} a_1 & 1-a_2 & 0 & \dots & & & 0 \\ 0 & a_2 & 1-a_3 & 0 & \dots & & 0 \\ 0 & 0 & a_3 & 1-a_4 & 0 & \dots & 0 \\ \vdots & & & & & & \\ 0 & \dots & & & 0 & a_{n+j-1} & 1-a_{n+j} \end{bmatrix}. \tag{6.10}$$

Έτσι, η εξίσωση (5.1) μπορεί να γραφτεί σε μητρωική μορφή ώστε να αποδώσει την ακολουθία των μεταβλητών ελέγχου που κατασκευάζονται κατά την διαδικασία προσθήκης κόμβων ως εξής

$$\overline{\mathbf{P}}^{j+1} = \left(\mathbf{C}^j\right)^T \overline{\mathbf{P}}^j \tag{6.11}$$

όπου $\overline{\mathbf{P}}^1 = \mathbf{P}$. Το τελικό σύνολο σημείων ελέγχου $\overline{\mathbf{P}}^{m+1}$ καθορίζει τα στοιχεία Bézier της αποσύνθεσης. Θέτοντας $\mathbf{P}^b = \overline{\mathbf{P}}^{m+1}$ και ορίζοντας $\mathbf{C}^T = (\mathbf{C}^m)^T(\mathbf{C}^{m-1})^T \dots (\mathbf{C}^1)^T$, η σχέση η οποία σχετίζει τα νέα σημεία ελέγχου Bézier με τα αρχικά της βάσης B-spline/NURBS είναι η εξής

$$\mathbf{P}^b = \mathbf{C}^T\mathbf{P}. \tag{6.12}$$

Δεδομένου ότι το μητρώο $\mathbf{P}$ έχει διάσταση $n \times d$, το μητρώο $\mathbf{C}$ έχει διάσταση $n \times (n+m)$ και το $\mathbf{P}^b$ διάσταση $(n+m) \times d$.

Η καμπύλη B-spline $\mathbf{C}(\xi)$ γράφεται ως προς τη βάση B-spline σε μητρωική μορφή ως εξής

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\mathbf{P}_i = \mathbf{P}^T\mathbf{N}(\xi) \tag{6.13}$$

Καθώς η εισαγωγή κόμβων δεν επιφέρει γεωμετρική αλλαγή στην καμπύλη, έτσι και η αποσύνθεση Bézier δεν επηρεάζει την καμπύλη $\mathbf{C}$, δηλαδή ισχύει

$$\mathbf{C}(\bar{\xi}) = \left(\mathbf{P^b}\right)^T\mathbf{B}(\bar{\xi}) = (\mathbf{C}^T\mathbf{P})^T\mathbf{B}(\bar{\xi}) = \mathbf{P}^T\mathbf{C}\mathbf{B}(\bar{\xi}) = \mathbf{P}^T\mathbf{N}(\bar{\xi}) \tag{6.14}$$

Επομένως, οι νέες συναρτήσεις Bernstein σχετίζονται με τις αρχικές συναρτήσεις B-spline/NURBS σύμφωνα με την παρακάτω σχέση

$$\mathbf{N}(\bar{\xi}) = \mathbf{C}\mathbf{B}(\bar{\xi}). \tag{6.15}$$

Το μητρώο $\mathbf{C}$ καλείται *τελεστής εξαγωγής Bézier* (Bézier extraction operator). Τονίζεται, δε, ότι το μοναδικό δεδομένο που απαιτείται για την κατασκευή του $\mathbf{C}$ είναι το κομβοδιάνυσμα. Με άλλα λόγια, ο τελεστής εξαγωγής εξαρτάται αποκλειστικά και μόνο από την παραμετροποίηση του παραμετρικού χωρίου και είναι ανεξάρτητος των σημείων ελέγχου και των συναρτήσεων βάσης. Δηλαδή, ο τελεστής εξαγωγής είναι ίδιος για τις συναρτήσεις B-spline και NURBS που μοιράζονται το ίδιο κομβοδιάνυσμα.

Στον ορισμό των συναρτήσεων βάσης NURBS, η συνάρτηση βάρους $W(\xi)$, η οποία υπεσέρχεται στον παρονομαστή της εξίσωσης, μπορεί να γραφτεί ως

$$W(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) w_i = \mathbf{w}^T \mathbf{N}(\xi) = \mathbf{w}^T \mathbf{C}\mathbf{B}(\xi) = (\mathbf{C}^T \mathbf{w})^T \mathbf{B}(\xi) = \left(\mathbf{w}^b\right)^T \mathbf{B}(\xi) = W^b(\xi) \quad (6.16)$$

όπου $\mathbf{w}^b = \mathbf{C}^T \mathbf{w}$, τα βάρη που σχετίζονται με τις συναρτήσεις βάσης Bézier, δοσμένα σε ένα διάνυσμα-στήλη.

Αντικαθιστώντας την (6.16) στην (2.20), οι συναρτήσεις NURBS γράφονται συναρτήσει του τελεστή εξαγωγής Bézier ως εξής

$$\mathbf{R}(\bar{\xi}) = \frac{1}{W^b(\xi)} \mathbf{W}\mathbf{C}\mathbf{B}(\bar{\xi}). \quad (6.17)$$

Όπως και στην εισαγωγή κόμβων, η εξαγωγή των ανανεωμένων σημείων Bézier υλοποιείται κατευθείαν στην καμπύλη B-spline η οποία ορίζει την καμπύλη NURBS. Γεωμετρικά, η παραπάνω διαδικασία ερμηνεύεται ως η προβολή των σημείων ελέγχου NURBS στο χώρο $\mathbb{R}^{d+1}$ [19] και εφαρμόζοντας, στη συνέχεια, τον τελεστή εξαγωγής Bézier στα σημεία ελέγχου της καμπύλης B-spline και εν τέλει προβάλλοντας πίσω στο χώρο $\mathbb{R}^d$ για τον υπολογισμό της σχέσης μεταξύ των σημείων ελέγχου Bézier και των σημείων ελέγχου NURBS, έχουμε

$$\mathbf{P}^b = \left(\mathbf{W}^b\right)^{-1} \mathbf{C}^T \mathbf{W}\mathbf{P}, \quad (6.18)$$

όπου $\mathbf{W}^b$ είναι τα βάρη Bézier διατεταγμένα σε ένα διαγώνιο πίνακα, δηλαδή

$$\mathbf{W}^b = \begin{bmatrix} w_1^b & & & \\ & w_2^b & & \\ & & \ddots & \\ & & & w_n^b \end{bmatrix} \quad (6.19)$$

Πολλαπλασιάζοντας την εξίσωση (6.18) από αριστερά με το $\mathbf{W}^b$, έχουμε

$$\mathbf{W}^b \mathbf{P}^b = \mathbf{C}^T \mathbf{W}\mathbf{P}, \quad (6.20)$$

Αξιοποιώντας την παραπάνω σχέση και την (2.21), η καμπύλη NURBS, γράφεται συναρτήσει των πολυωνύμων Bernstein ως

$$\begin{aligned} \mathbf{C}(\xi) &= \sum_{i}^{n} \mathbf{P}_i R_i(\xi) = \mathbf{P}^T \mathbf{R}(\xi) = \frac{1}{W(\xi)} \mathbf{P}^T \mathbf{W}\mathbf{N}(\xi) \\ &= \frac{1}{W^b(\xi)} \mathbf{P}^T \mathbf{W}\mathbf{C}\mathbf{B}(\xi) = \frac{1}{W^b(\xi)} (\mathbf{C}^T \mathbf{W}\mathbf{P})^T \mathbf{B}(\xi) \\ &= \frac{1}{W^b(\xi)} \left(\mathbf{W}^b \mathbf{P}^b\right)^T \mathbf{B}(\xi) \end{aligned} \quad (6.21)$$

6.4 Ο τοπικός τελεστής Bézier και το στοιχείο Bézier

Αν υπολογιστεί ο τελεστής εξαγωγής Bézier **C** για το παράδειγμα του Σχήμα 6-1, χρησιμοποιώντας τις εξισώσεις (6.10) και (6.11), η εξίσωση (6.15) γράφεται ως

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{7}{12} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{7}{12} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \\ B_8 \\ B_9 \\ B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix}. \qquad (6.22)$$

Η εξαγωγή Bézier δημιουργεί ένα στοιχείο Bézier για κάθε μη μηδενικό διάστημα (κομβοδιάστημα) του αρχικού κομβοδιανύσματος. Επομένως, σε κάθε κομβοδιάστημα, οι αρχικές συναρτήσεις NURBS μπορούν να εκφραστούν ως γραμμικός συνδυασμός των συναρτήσεων βάσης του στοιχείου Bézier που αντιστοιχεί στο εκάστοτε κομβοδιάστημα. Για παράδειγμα, στο πρώτο κομβοδιάστημα [0,1), αντιστοιχούν οι συναρτήσεις NURBS N_1, N_2, N_3, N_4, οι οποίες μπορούν να αναπαρασταθούν ως γραμμικός συνδυασμός των συναρτήσεων Bézier B_1, B_2, B_3, B_4. Οι εν λόγω συναρτήσεις βάσης σχετίζονται γραμμικά σύμφωνα με την παρακάτω εξίσωση

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} = \begin{Bmatrix} N_1^1 \\ N_2^1 \\ N_3^1 \\ N_4^1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} B_1^1 \\ B_2^1 \\ B_3^1 \\ B_4^1 \end{Bmatrix} \qquad (6.23)$$

στην οποία ο άνω δείκτης αντιστοιχεί στο κομβοδιάστημα ή (ισοδύναμα) στον αριθμό του τρέχοντος στοιχείου.

Για τα υπόλοιπα τρία κομβοδιαστήματα [1,2), [2,3), [3,4), έχουμε τις παρακάτω σχέσεις

$$\begin{Bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{Bmatrix} = \begin{Bmatrix} N_1^2 \\ N_2^2 \\ N_3^2 \\ N_4^2 \end{Bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ \frac{7}{12} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{Bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ \frac{7}{12} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} B_1^2 \\ B_2^2 \\ B_3^2 \\ B_4^2 \end{Bmatrix} \tag{6.24}$$

$$\begin{Bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix} = \begin{Bmatrix} N_1^3 \\ N_2^3 \\ N_3^3 \\ N_4^3 \end{Bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{Bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{Bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{Bmatrix} B_1^3 \\ B_2^3 \\ B_3^3 \\ B_4^3 \end{Bmatrix} \tag{6.25}$$

$$\begin{Bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} = \begin{Bmatrix} N_1^4 \\ N_2^4 \\ N_3^4 \\ N_4^4 \end{Bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{7}{12} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{7}{12} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_1^4 \\ B_2^4 \\ B_3^4 \\ B_4^4 \end{Bmatrix} \tag{6.26}$$

Για τον υπολογισμό των τοπικών τελεστών εξαγωγής Bézier, ορίζονται οι τοπικές συναρτήσεις βάσης για κάθε στοιχείο ως εξής:

$$\begin{Bmatrix} N_1^1 \\ N_2^1 \\ N_3^1 \\ N_4^1 \end{Bmatrix} = \begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix}, \begin{Bmatrix} N_1^2 \\ N_2^2 \\ N_3^2 \\ N_4^2 \end{Bmatrix} = \begin{Bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{Bmatrix},$$
$$\begin{Bmatrix} N_1^3 \\ N_2^3 \\ N_3^3 \\ N_4^3 \end{Bmatrix} = \begin{Bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix}, \begin{Bmatrix} N_1^4 \\ N_2^4 \\ N_3^4 \\ N_4^4 \end{Bmatrix} = \begin{Bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} \tag{6.27}$$

και

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

$$\begin{Bmatrix} B_1^3 \\ B_2^3 \\ B_3^3 \\ B_4^3 \end{Bmatrix} = \begin{Bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{Bmatrix}, \begin{Bmatrix} B_1^4 \\ B_2^4 \\ B_3^4 \\ B_4^4 \end{Bmatrix} = \begin{Bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix}, \tag{6.28}$$

Επομένως, έχοντας ορίσει τις τοπικές συναρτήσεις βάσης NURBS και Bernstein αντίστοιχα, οι τοπικοί τελεστές εξαγωγής Bézier θα είναι οι εξής:

$$\mathbf{C}^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix}, \mathbf{C}^2 = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ \frac{7}{12} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix}$$

$$\mathbf{C}^3 = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix}, \mathbf{C}^4 = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{7}{12} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.29}$$

Επομένως, η τοπική διατύπωση της εξίσωσης (6.15) ανά στοιχείο NURBS είναι

$$\mathbf{N}^e(\tilde{\xi}) = \mathbf{C}^e \mathbf{B}^e(\tilde{\xi}), \tag{6.30}$$

όπου e ο αύξων αριθμός του στοιχείου. Σημειώνεται ότι η διάσταση του $\mathbf{C}^e$ είναι $(p+1) \times (p+1)$.

Επιπλέον, η εξίσωση (6.17) μπορεί να γραφτεί ως προς τις τοπικές συναρτήσεις NURBS κάθε στοιχείου e ως

$$\mathbf{R}^e(\xi) = \frac{1}{W^b(\xi)} \mathbf{W}^e \mathbf{C}^e \mathbf{B}^e(\xi) \tag{6.31}$$

όπου

$$W^b(\xi) = \sum_{1}^{(p+1)^{d_p}} B_{i,p}(\xi) w_i^b \tag{6.32}$$

με d_p, το πλήθος των παραμετρικών διαστάσεων. Εφαρμόζοντας τον κανόνα του πηλίκου στην εξίσωση (6.31), λαμβάνονται οι παράγωγοι των τοπικών συναρτήσεων NURBS ως εξής

$$\frac{\partial R^e(\xi)}{\partial \xi} = \mathbf{W}^e\mathbf{C}^e \frac{\partial}{\partial \xi}\left(\frac{1}{W^b(\xi)}\mathbf{B}^e(\xi)\right) = \mathbf{W}^e\mathbf{C}^e\left(\frac{1}{W^b(\xi)}\frac{\partial \mathbf{B}^e(\xi)}{\partial \xi} - \frac{\partial W^b(\xi)}{\partial \xi}\frac{\mathbf{B}^e(\xi)}{\left(W^b(\xi)\right)^2}\right) \quad (6.33)$$

όπου η παράγωγος της συνάρτησης βάρους υπολογίζεται από τη σχέση

$$\frac{\partial W^b(\xi)}{\partial \xi} = \sum_{i=1}^{(p+1)^{d_p}} B'_{i,p}(\xi) w_i^b \quad (6.34)$$

Η σχέση μεταξύ των τοπικών σημείων ελέγχου Bézier κάθε στοιχείου και των τοπικών σημείων ελέγχου NURBS είναι παρόμοια με εκείνη των καθολικών σημείων ελέγχου (6.18),

$$\mathbf{P}^{b,e} = \left(\mathbf{W}^{b,e}\right)^{-1}\mathbf{C}^{e^T}\,\mathbf{W}^e\mathbf{P}^e, \quad (6.35)$$

όπου $\mathbf{W}^{b,e}$ είναι διαγώνιος πίνακας με διαγώνιο τα τοπικά βάρη Bézier,

$$\mathbf{W}^{b,e} = \begin{bmatrix} w_1^{b,e} & & & \\ & w_2^{b,e} & & \\ & & \ddots & \\ & & & w_n^{b,e} \end{bmatrix} \quad (6.36)$$

και

$$\mathbf{w}^{b,e} = (\mathbf{C}^e)^T\mathbf{w}^e \quad (6.37)$$

6.5 Ο διδιάστατος τελεστής εξαγωγής

Στη διδιάστατη περίπτωση, όπου $\boldsymbol{\xi} = (\xi, \eta)$, ο τοπικός διδιάστατος τελεστής εξαγωγής Bézier ορίζεται ως το τανυστικό γινόμενο των μονοδιάστατων τελεστών,

$$\mathbf{C}^e = \mathbf{C}_\eta^j \otimes \mathbf{C}_\xi^i = \begin{bmatrix} C_{\eta,11}^j\mathbf{C}_\xi^i & C_{\eta,12}^j\mathbf{C}_\xi^i & \cdots \\ C_{\eta,21}^j\mathbf{C}_\xi^i & C_{\eta,22}^j\mathbf{C}_\xi^i & \\ \vdots & & \ddots \end{bmatrix} \quad (6.38)$$

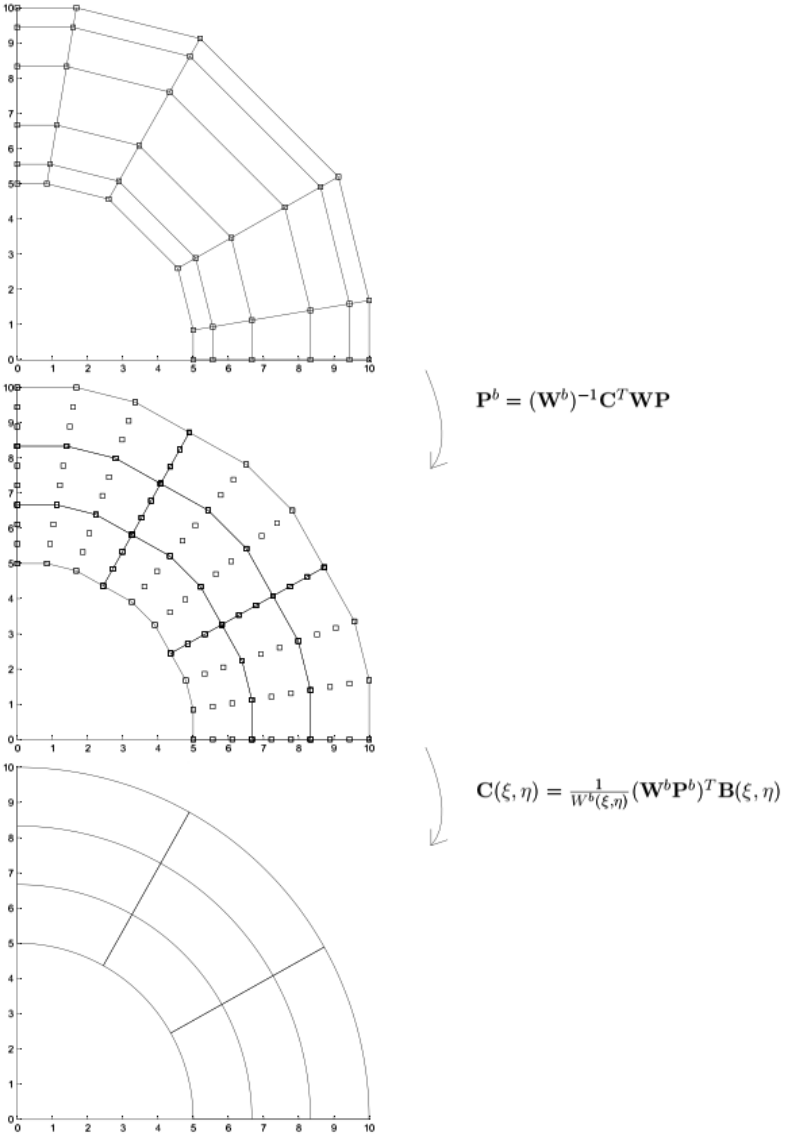
όπου $\mathbf{C}_\xi^i$ και $\mathbf{C}_\eta^j$ είναι ο i-οστός και j-οστός μονοδιάστατος τοπικός τελεστής εξαγωγής για την κατεύθυνση ξ και η, αντίστοιχα, με $i = 1,2,\dots,n$, $j = 1,2,\dots,m$ και $e = 1,2,\dots,n \times m$.

Γνωρίζοντας, πλέον, τους διδιάστατους τοπικούς τελεστές Bézier, τα σημεία ελέγχου Bézier κάθε στοιχείου μπορούν να υπολογιστούν από τη σχέση (6.35). Επιπλέον, το φυσικό πλέγμα Bézier προσδιορίζεται από τη σχέση:

$$\mathbf{C}(\tilde{\xi}, \tilde{\eta}) = \frac{1}{W^b(\xi)}\left(\mathbf{W}^b\mathbf{P}^b\right)^T\mathbf{B}(\tilde{\xi}, \tilde{\eta}) \quad (6.39)$$

όπου $\tilde{\boldsymbol{\xi}} = (\tilde{\xi}^1, \tilde{\xi}^2) = (\tilde{\xi}, \tilde{\eta})$ είναι το σύστημα συντεταγμένων, το οποίο αποδόθηκε στο γονικό στοιχείο (parent element).

Η απεικόνιση του πλέγματος ελέγχου της βάσης NURBS στο πλέγμα ελέγχου της βάσης Bézier παρουσιάζεται στο Σχήμα 6-3. Τονίζεται ότι τα στοιχεία Bézier ισοδυναμούν με τα στοιχεία της βάσης NURBS, καθώς η αποσύνθεση Bézier αφήνει αναλλοίωτη τη γεωμετρία, με τη μόνη διαφορά να βρίσκεται στην αύξηση των σημείων ελέγχου. [6]



Σχήμα 6-3 Από το πλέγμα ελέγχου NURBS στο πλέγμα ελέγχου Bézier στο φυσικό πλέγμα Bézier [6]

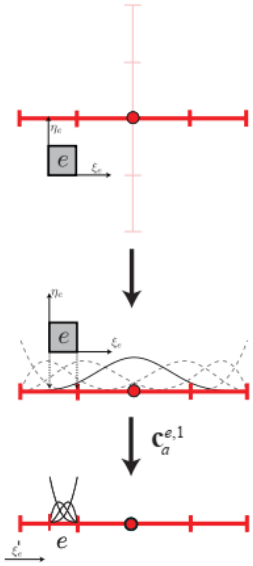
6.6 Ο τελεστής εξαγωγής Bézier για T-Spline

Γνωρίζουμε ότι στα NURBS, ο παραμετρικός χώρος είναι ένας καρτεσιανός χώρος, στον οποίο οι συναρτήσεις βάσης NURBS υπολογίζονται ως το τανυστικό γινόμενο των συναρτήσεων βάσης κάθε διάστασης. Στη βάση T-spline, ο παραμετρικός χώρος δεν υφίσταται ως καθολικό πεδίο υπό μορφή καρτεσιανού επιπέδου. Παρ' όλα αυτά, μπορεί να οριστεί ένα τοπικό καρτεσιανό επίπεδο για κάθε

συνάρτηση βάσης T-spline με σημείο αφετηρίας το δεσμό της συνάρτησης T-spline. Το επίπεδο αυτό καλείται *τοπικό πεδίο συνάρτησης βάσης* (local basis function domain) και ορίζεται για κάθε συνάρτηση T-spline (Κεφάλαιο 3.3). Γίνεται, λοιπόν, σαφές ότι οι τοπικοί τελεστές εξαγωγής για κάθε στοιχείο δεν μπορούν υπολογιστούν ως τα τανυστικά γινόμενα των μονοδιάστατων τελεστών εξαγωγής, όπως συμβαίνει στη βάση NURBS. Αντιθέτως, ο υπολογισμός του τοπικού τελεστή του στοιχείου πραγματοποιείται ξεχωριστά για κάθε μη μηδενική συνάρτηση του στοιχείου. Έτσι, το αποτέλεσμα της τοπικής εξαγωγής Bézier ανά συνάρτηση βάσης T-spline δεν είναι ένας πίνακας, όπως στη βάση NURBS, αλλά μόνο μια γραμμή, η οποία αντιστοιχεί στην συνάρτηση για την οποία πραγματοποιήθηκε η εξαγωγή. Με την υλοποίηση της εξαγωγής Bézier για κάθε συνάρτηση T-spline που υποστηρίζεται στο στοιχείο, δομείται ο τελικός πίνακας της τοπικής εξαγωγής Bézier, δηλαδή ο τοπικός τελεστής εξαγωγής Bézier.

Η δεύτερη διαφορά ανάμεσα στην Αποσύνθεση Bézier σε συναρτήσεις T-spline και συναρτήσεις NURBS/B-spline είναι ότι σε αντίθεση με τις συναρτήσεις NURBS, τα κομβοδιανύσματα στα T-spline δεν είναι κατά κανόνα ανοιχτά. Επομένως, πριν την έναρξη της εξαγωγής στα T-spline, απαιτείται η κατασκευή των *επεκτεταμένων κομβοδιανυσμάτων* (extended knot vectors). Τα επεκτεταμένα κομβοδιανύσματα προκύπτουν από την επανάληψη των ακραίων κομβικών τιμών των αρχικών κομβοδιανυσμάτων μέχρι η πολλαπλότητα των κόμβων να γίνει ίση με $p + 1$.

Τέλος, λόγω των διασταυρώσεων T σε πλέγματα T-spline, είναι πιθανό να απαιτείται η επιπλέον προσθήκη κόμβου στο κομβοδιάνυσμα κάθε κατεύθυνσης για τον σχηματισμό του στοιχείου εξαγωγής. Αυτό το ενδεχόμενο δεν υφίσταται σε πλέγματα B-spline/NURBS, όπου τα κομβοδιαστήματα ταυτίζονται με τα στοιχεία του πλέγματος. Στο Σχήμα 6-4 απεικονίζεται η εξαγωγή Bézier μιας συνάρτησης T-spline στην κατεύθυνση ξ του στοιχείου e του πλέγματος. Για την υλοποίηση της εξαγωγής, προστίθεται στο κομβοδιάνυσμα ένας επιπλέον κόμβος με πολλαπλότητα p, του οποίου η τιμή αντιστοιχεί στη συντεταγμένη x του στοιχείου που δεν περιλαμβάνεται στο αρχικό τοπικό κομβοδιάνυσμα.

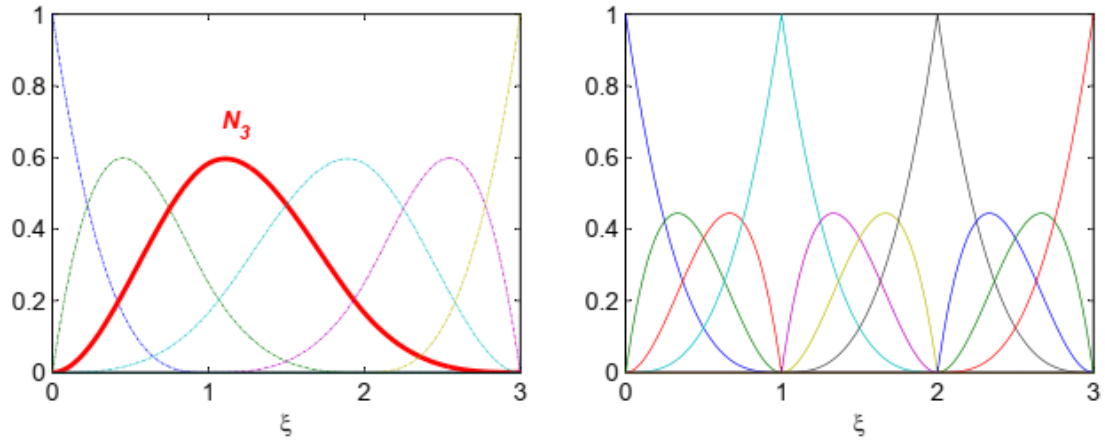


Σχήμα 6-4 Εφαρμογή της εξαγωγής Bézier στην κατεύθυνση ξ μιας διδιάστατης συνάρτησης T-spline. Επισημαίνεται η αναγκαιότητα προσθήκης επιπλέον κόμβου πολλαπλότητας p στο τοπικό κομβοδιάνυσμα για την κατασκευή του στοιχείου e στην κατεύθυνση ξ. [11]

Όπως και στα NURBS, για να αποσυνθέσουμε τη βάση συναρτήσεων στα κατά Bézier στοιχεία της, δηλαδή για να επιτύχουμε την Αποσύνθεση Bézier (Bézier Decomposition), όλοι οι εσωτερικοί κόμβοι επαναλαμβάνονται μέχρι να αποκτήσουν πολλαπλότητα ίση με p. Με αυτόν τον τρόπο, οι συναρτήσεις βάσης NURBS θα είναι C^0-συνεχείς στο σύνορο των στοιχείων και εντός κάθε στοιχείου θα περιγράφονται ως ο γραμμικός συνδυασμός των πολυώνυμων Bernstein βαθμού p. Θεωρητικά, για τον πλήρη διαχωρισμό των στοιχείων Bézier, η πολλαπλότητα των εσωτερικών κόμβων θα έπρεπε να είναι $p + 1$. Ωστόσο, η πολλαπλότητα p είναι επαρκής για την αναπαράσταση των πολυωνύμων Bernstein.

Η εξαγωγή Bézier για T-spline απεικονίζει τις συναρτήσεις T-spline πάνω σε κάθε στοιχείο ως γραμμικό συνδυασμό των πολυωνυμικών συναρτήσεων Bernstein των στοιχείων Bézier. Επισημαίνεται ότι, το στοιχείο Bézier ορίζεται στο φυσικό χώρο από τα μη μηδενικά κομβικά διαστήματα του T-spline.

Στο Σχήμα 6-5$^{\alpha}$ απεικονίζεται η συνάρτηση βάσης T-spline N_3 του τοπικού κομβοδιανύσματος $\Xi = \{0,0,1,2,3\}$ με $p = 3$. Οι λεπτές διακεκομμένες γραμμές είναι οι πρόσθετες συναρτήσεις βάσης που σχηματίζονται από το επεκτεταμένο κομβοδιάνυσμα $\bar{\Xi} = \{0,0,0,0,1,2,3,3,3,3\}$. Επομένως, χρησιμοποιώντας το κομβοδιάνυσμα $\bar{\Xi}$, οι τελεστές εξαγωγής Bézier μπορούν να υπολογιστούν με παρόμοιο τρόπο όπως στη βάση NURBS για να λάβουμε τις συναρτήσεις βάσης των στοιχείων Bézier που φαίνονται στο Σχήμα 6-5$^{\beta}$. Οι τελεστές εξαγωγής Bézier θα ισούνται, επομένως, με τους τελεστές στην περίπτωση NURBS. [6]



(α) Οι συνάρτηση βάσης N_3 για το τοπικό κομβοδιάνυσμα $\Xi = \{0,0,1,2,3\}$ και οι πρόσθετες συναρτήσεις βάσης (λεπτές διακεκομμένες γραμμές) για το επεκτεταμένο κομβοδιάνυσμα $\bar{\Xi} = \{0,0,0,0,1,2,3,3,3,3\}$.

(β) Οι συναρτήσεις βάσης Bézier μετά την αποσύνθεση Bézier του κομβοδιανύσματος $\bar{\Xi} = \{0,0,0,0,1,2,3,3,3,3\}$.

Σχήμα 6-5 Αποσύνθεση Bézier σε μονοδιάστατη συνάρτηση βάσης T-spline. [6]

$$\begin{Bmatrix} N_1 \\ N_2 \\ \boldsymbol{N_3} \\ N_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ \mathbf{0} & \mathbf{0} & \mathbf{\frac{1}{2}} & \mathbf{\frac{7}{12}} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} \boldsymbol{B_1} \\ \boldsymbol{B_2} \\ \boldsymbol{B_3} \\ \boldsymbol{B_4} \end{Bmatrix} \tag{6.42}$$

$$\begin{Bmatrix} N_2 \\ \boldsymbol{N_3} \\ N_4 \\ N_5 \end{Bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ \mathbf{\frac{7}{12}} & \mathbf{\frac{2}{3}} & \mathbf{\frac{1}{3}} & \mathbf{\frac{1}{6}} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} \boldsymbol{B_4} \\ \boldsymbol{B_5} \\ \boldsymbol{B_6} \\ \boldsymbol{B_7} \end{Bmatrix} \tag{6.43}$$

$$\begin{Bmatrix} \boldsymbol{N_3} \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix} = \begin{bmatrix} \mathbf{\frac{1}{6}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{7}{12} \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{Bmatrix} \boldsymbol{B_7} \\ \boldsymbol{B_8} \\ \boldsymbol{B_9} \\ \boldsymbol{B_{10}} \end{Bmatrix} \tag{6.44}s$$

Σημειώνεται, ωστόσο, ότι μόνο οι γραμμές των τελεστών εξαγωγής με έντονη γραφή είναι απαραίτητες για την απεικόνιση των πολυωνύμων Bernstein των στοιχείων Bézier στην καθολική

συνάρτηση βάσης T-spline N_3. Επομένως, ένας αποδοτικός και οικονομικός αλγόριθμος υπολογισμού των τελεστών Bézier για T-spline δε θα πρέπει να υπολογίζει τις περίσσιες γραμμές του πίνακα, αλλά μόνο τη μία γραμμή που αναλογεί στην εκάστοτε συνάρτηση T-spline. [6]

Δεδομένου ότι μια καμπύλη T-spline δεν έχει κάποια ουσιαστική διαφορά από μια καμπύλη B-spline, το ενδιαφέρον των T-spline εστιάζεται στις πολυδιάστατες περιπτώσεις και δη στη διδιάστατη περίπτωση. Ο τοπικός διδιάστατος τελεστής για τα T-spline περιλαμβάνει μια γραμμή μήκους $(p+1)^2$ για κάθε συνάρτηση βάσης που υποστηρίζεται στο στοιχείο T-spline. Το μήκος της γραμμής αντιστοιχεί στο πλήθος των πολυωνύμων Bernstein του διδιάστατου στοιχείου, το οποίο είναι κοινό για όλα τα στοιχεία. Το πλήθος των συναρτήσεων βάσης T-spline που υποστηρίζονται σε κάθε στοιχείο, όμως, μπορεί να είναι $(p+1)^2$ ή και μεγαλύτερο, καθώς οι διασταυρώσεις T του T-mesh επιδρούν στη δομή του στοιχείου. Συνεπώς, το πλήθος των γραμμών στους τοπικούς τελεστές εξαγωγής για T-spline μπορεί σε κάποια στοιχεία να είναι μεγαλύτερο από το σταθερό $(p+1)^2$ που είχαμε στους τελεστές εξαγωγής για τα διδιάστατα NURBS. Επομένως, ο τελεστής εξαγωγής για T-spline μπορεί να χειριστεί και τους «κρεμαστούς κόμβους» (hanging nodes) της συμβατικής Ανάλυσης Πεπερασμένων Στοιχείων.

7 ΤΑΞΙΝΌΜΗΣΗ ΤΩΝ T-SPLINE

Για την εφαρμογή της Ισογεωμετρικής ανάλυσης στα T-spline, η γραμμική ανεξαρτησία των συναρτήσεων είναι επιτακτικής σημασίας. Η γραμμική εξάρτηση των συναρτήσεων συνεπάγεται ένα σύστημα εξισώσεων μη αντιστρέψιμο και η επίλυση αυτή της δυσκολίας οδήγησε στην καθιέρωση της απαίτησης της γραμμικής ανεξαρτησίας των συναρτήσεων ανάμειξης. Όπως έχει ήδη αναφερθεί στο Κεφάλαιο 4, υπάρχει ένα περιορισμένο υποσύνολο των T-spline, για το οποίο γνωρίζουμε ότι ισχύει η γραμμική ανεξαρτησία, τα κατάλληλα-προς-ανάλυση T-spline, στα οποία οι κάθετες επεκτάσεις των διασταυρώσεων T δεν τέμνονται. Επιπλέον, αξίζει να τονιστεί ότι η γραμμική ανεξαρτησία των πολυωνυμικών συναρτήσεων T-spline είναι ισοδύναμη με την γραμμική ανεξαρτησία των ρητών συναρτήσεων T-spline, λαμβάνοντας ως δεδομένο ότι τα βάρη είναι θετικοί αριθμοί. Δηλαδή, η επιλογή των (αυστηρά θετικών) βαρών δεν επηρεάζει την γραμμική ανεξαρτησία των ρητών συναρτήσεων. [15]

Όσον αφορά την ιδιότητα της διαμέρισης της μονάδας, είναι γνωστό ότι τόσο οι συναρτήσεις B-spline όσο και οι συναρτήσεις NURBS επαληθεύουν την άθροιση των συναρτήσεων στη μονάδα σε κάθε παραμετρική θέση, παρ' όλα αυτά, δεν ισχύει εκ των προτέρων το ίδιο και για τις συναρτήσεις T-spline (Κεφάλαιο 4.5).

Για την απάντηση των ερωτημάτων που αφορούν τη γραμμική εξάρτηση και τη διαμέριση της μονάδας από την πολυωνυμική βάση T-spline μπορεί να αξιοποιηθεί ο τελεστής εξαγωγής Bézier, σύμφωνα με το [14], όπως αναλύεται παρακάτω.

7.1 Ταξινόμηση των T- με τη χρήση του τελεστή εξαγωγής Bézier

Στο παρόν κεφάλαιο θα διερευνήσουμε την ταξινόμηση των T-spline ως προς την γραμμική ανεξαρτησία που παρουσιάζουν οι συναρτήσεις T-spline. Επιπλέον, θα ταξινομήσουμε τα πλέγματα T-spline ως προς την ικανότητα ή μη των πολυωνυμικών συναρτήσεων T-spline να ικανοποιήσουν την ιδιότητα της διαμέρισης της μονάδας. Το κριτήριο και για τις δύο ταξινομήσεις απορρέει από τους τοπικούς τελεστές εξαγωγής Bézier. Με τις παραπάνω κατηγοριοποιήσεις, η ανάλυση του T-mesh αποσυνδέεται από την έννοια του "analysis-suitable admissible T-mesh", η οποία απέκλειε τοπολογίες T-mesh, οι οποίες θα μπορούσαν να χρησιμοποιηθούν στην ανάλυση. Η μέθοδος ταξινόμησης των T-spline, σύμφωνα με τον May κ.ά. [14], μπορεί να εφαρμοστεί σε πλέγματα T-spline αυθαίρετου βαθμού, σε πλέγματα με εξαιρετικά σημεία (extraordinary points) και σε τρισδιάστατα πλέγματα. Η μοναδική προϋπόθεση είναι να γνωρίζουμε τους τοπικούς τελεστές εξαγωγής Bézier.

Αν η εξαγωγή Bézier για το δεσμό i εφαρμοστεί σε όλα τα στοιχεία E του πλέγματος, τότε ο τελεστής εξαγωγής Bézier για το συγκεκριμένο δεσμό υπολογίζεται ως

$$\underline{\boldsymbol{C}}^i = \begin{bmatrix} \underline{\boldsymbol{C}}_1^i \\ \vdots \\ \underline{\boldsymbol{C}}_E^i \end{bmatrix} \tag{7.1}$$

Τότε, ο *καθολικός τελεστής εξαγωγής Bézier* για όλες τους δεσμούς n δίνεται από τη σχέση:

$$\underline{\underline{\mathbf{C}}} = \begin{bmatrix} \underline{\boldsymbol{C}}^{1^T} \\ \vdots \\ \underline{\boldsymbol{C}}^{n^T} \end{bmatrix}. \tag{7.2}$$

Σημειώνεται δε ότι ο καθολικός τελεστής εξαγωγής Bézier όπως ορίζεται εδώ δεν ταυτίζεται με τον τελεστή εξαγωγής του Κεφαλαίου 6. Για τους σκοπούς της ταξινόμησης των T-spline ο ορισμός του τελεστή απλοποιείται και ορίζεται ως ένας πίνακας στον οποίο εναποτίθενται οι τοπικοί πίνακες εξαγωγής, χωρίς, όμως, να λαμβάνεται υπόψιν η αντιστοίχιση των νέων συναρτήσεων Bernstein στα επιμέρους στοιχεία, δηλαδή χωρίς να υπολογίζεται το μητρώο IEN του εκλεπτυσμένου πλέγματος.

7.1.1 Ταξινόμηση των T-spline ως προς τη γραμμική εξάρτηση

Ο καθολικός τελεστής εξαγωγής Bézier κατατάσσει τα πλέγματα T-spline σε τρεις κατηγορίες, ανάλογα με τον τύπο γραμμικής εξάρτησης που παρουσιάζουν οι συναρτήσεις μεταξύ τους:

i. καθολική γραμμική ανεξαρτησία,
ii. τοπική γραμμική ανεξαρτησία με μη τετραγωνικό πίνακα C_e,
iii. τοπική γραμμική ανεξαρτησία με τετραγωνικό πίνακα C_e.

7.1.1.1 Καθολική γραμμική ανεξαρτησία

Ένα πλέγμα T-spline με n δεσμούς έχει καθολικώς γραμμικά ανεξάρτητες συναρτήσεις, αν

$$\mathrm{rank}\left(\underline{\underline{\mathbf{C}}}\right) = n. \tag{7.3}$$

Σημειώνεται, επίσης, ότι το μέγεθος του καθολικού τελεστή εξαγωγής Bézier $\underline{\underline{\mathbf{C}}}$ είναι

$$\mathrm{size}\left(\underline{\underline{\mathbf{C}}}\right) = n \times \left(E \times \prod_{l=1}^{d} (p_l + 1) \right). \tag{7.4}$$

Ένα πλέγμα T-spline με καθολικώς γραμμικά εξαρτημένες συναρτήσεις ανάμειξης δεν μπορεί να χρησιμοποιηθεί στην ανάλυση, καθώς αυτή η εξάρτηση θα οδηγούσε σε ένα σύστημα εξισώσεων το οποίο δε θα μπορούσε να επιλυθεί.

7.1.1.2 Τοπική γραμμική ανεξαρτησία

Κατ' αντιστοιχία με την καθολική γραμμική ανεξαρτησία, ένα πλέγμα T-spline είναι τοπικώς γραμμικά ανεξάρτητο, αν

$$\mathrm{rank}\left(\underline{\underline{\mathbf{C}}}_e\right) = n_e \text{ για } e = 1, \dots, E, \tag{7.5}$$

όπου C_e, ο τοπικός τελεστής εξαγωγής Bézier και n_e, το πλήθος των συναρτήσεων με φορέα στο στοιχείο e.

Το μέγεθος του τοπικού τελεστή εξαγωγής Bézier, για $e = 1, \dots, E$ είναι

$$\text{size}(\underline{\underline{\mathbf{C}}}_e) = n_e \times \left(\prod_{l=1}^{d}(p_l + 1)\right). \tag{7.6}$$

7.1.1.3 Τοπική γραμμική ανεξαρτησία με τετραγωνικό C_e

Ορίζεται, επίσης, το υποσύνολο των τοπικώς γραμμικά ανεξάρτητων πλεγμάτων T-spline για τα οποία επαληθεύεται, επιπλέον, η ακόλουθη συνθήκη για κάθε στοιχείο e του T-spline:

$$\text{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{l=1}^{d}(p_l + 1) = n_e \text{ για } e = 1, \dots, E \tag{7.7}$$

Σε αυτή την περίπτωση, ισχύει

$$\text{size}(\underline{\underline{\mathbf{C}}}_e) = \left(\prod_{l=1}^{d}(p_l + 1)\right) \times \left(\prod_{l=1}^{d}(p_l + 1)\right). \tag{7.8}$$

Διαπιστώνουμε ότι η εξίσωση (7.7) συνεπάγεται την (7.5) κι αντίστοιχα (7.5) συνεπάγεται την εξίσωση (7.3), επομένως, η τοπική γραμμική ανεξαρτησία οδηγεί εγγενώς στην καθολική γραμμική ανεξαρτησία των συναρτήσεων.

7.1.2 Ταξινόμηση των T-spline ως προς τη διαμέριση της μονάδας

7.1.2.1 Η ιδιότητα της διαμέρισης της μονάδας στις ρητές συναρτήσεις R_i

Η πολυδιάστατη ρητή συνάρτηση T-spline για τον κόμβο i περιγράφεται από την εξίσωση

$$R_i\left(\underline{\xi}\right) = \frac{w_i N_i\left(\underline{\xi}\right)}{\sum_{j=1}^{n} w_j N_j\left(\underline{\xi}\right)}, \tag{7.9}$$

όπου w_i, ο συντελεστής βάρους που σχετίζεται με τον κόμβο i. Παρατηρούμε, λοιπόν, ότι οι ρητές συναρτήσεις R_i διαμερίζουν πάντα τη μονάδα, καθώς το άθροισμα όλων των R_i είναι μονάδα σε κάθε σημείο $\boldsymbol{\xi}$ του παραμετρικού χωρίου.

7.1.2.2 Η ιδιότητα της διαμέρισης της μονάδας στις πολυωνυμικές συναρτήσεις N_i

Τα πλέγματα T-spline μπορούν να ταξινομηθούν, σύμφωνα με την ιδιότητα της διαμέρισης της μονάδας των πολυωνυμικών συναρτήσεων N_i ως εξής [7]:

Έστω

$$\sum_{i=1}^{n} \beta_i N_i\left(\underline{\xi}\right) = 1 \tag{7.10}$$

τότε θα έχουμε

i. κανονικό (standard) πλέγμα T-spline, αν $\beta_i = 1$, για κάθε $i = 1, \dots, n$
ii. ημι-κανονικό πλέγμα (semi-standard) T-spline, αν $\beta_i = 1$, για τουλάχιστον ένα i
iii. μη κανονικό (non-standard) πλέγμα T-spline, αν δεν υπάρχει λύση για το β_i

Επίσης, σημειώνεται ότι η ιδιότητα της διαμέρισης μονάδας ικανοποιείται στην περίπτωση κανονικού πλέγματος με τη χρήση των πολυωνυμικών συναρτήσεων N_i, ενώ σε πλέγμα T-spline οποιουδήποτε άλλου τύπου, απαιτείται η χρήση των ρητών συναρτήσεων R_i για να ικανοποιείται η εν λόγω ιδιότητα.

Η παραπάνω εξίσωση οδηγεί στο σύστημα

$$\underline{\underline{\mathbf{C}}}^T \underline{\boldsymbol{\beta}} = \underline{\mathbf{1}}, \tag{7.11}$$

η λύση $\underline{\boldsymbol{\beta}}$ του οποίου καθορίζει σε ποια κατηγορία ανήκει το πλέγμα T-spline. Αν το $\underline{\boldsymbol{\beta}}$ είναι το μοναδιαίο διάνυσμα, το πλέγμα T-spline είναι κανονικό, ενώ αν στο διάνυσμα υπάρχει τιμή διάφορη της μονάδας, το πλέγμα είναι ημι-κανονικό. Τέλος, αν το σύστημα δεν έχει λύση, το πλέγμα είναι μη κανονικό.

Στην περίπτωση ενός κανονικού πλέγματος T-spline, έχουμε $\sum_{j=1}^{n} N_j\left(\underline{\boldsymbol{\xi}}\right) = 1$, δηλαδή αν $w_j = 1$, ο παρονομαστής $\sum_{j=1}^{n} w_j N_j\left(\underline{\boldsymbol{\xi}}\right)$ στον ορισμό των ρητών συναρτήσεων T-spline είναι μονάδα, κι ως εκ τούτου οι ρητές συναρτήσεις ταυτίζονται με τις πολυωνυμικές, $R_i = N_i$, δηλαδή το T-spline είναι μια πολυωνυμική επιφάνεια. Στην περίπτωση ενός ημι-κανονικού πλέγματος T-spline, το πλέγμα T-spline είναι ισοδύναμο με ένα κανονικό πλέγμα επιφάνειας NURBS. [5]

Όπως παρατηρήθηκε στο [14], η έννοια «κατάλληλο προς ανάλυση T-spline» με τον τρόπο που ορίστηκε στο [15] μπορεί να προκαλέσει σύγχυση, καθώς εν τέλει μέθοδοι ανάλυσης μπορούν να εφαρμοστούν σε κανονικά, ημι-κανονικά και μη κανονικά T-mesh., αρκεί να ικανοποιούνται οι παρακάτω προϋποθέσεις:

– οι πολυωνυμικές συναρτήσεις ανάμειξης N_i να είναι καθολικά γραμμικά ανεξάρτητες
– να επαληθεύεται η διαμέριση της μονάδας, ώστε να ικανοποιείται η αφινική συνδιακύμανση (affine covariance) και το “patch test”.

Σημειώνεται, επίσης, ότι η καθολική/τοπική γραμμική ανεξαρτησία των πολυωνυμικών συναρτήσεων N_i συνεπάγεται την καθολική/τοπική γραμμική ανεξαρτησία των ρητών συναρτήσεων ανάμειξης R_i. Έτσι, τα καθολικώς γραμμικά ανεξάρτητα ημι-κανονικά και μη κανονικά πλέγματα T-spline, μπορούν να χρησιμοποιηθούν στην ανάλυση, εφόσον χρησιμοποιηθούν οι ρητές συναρτήσεις βάσης R_i, καθώς για τις R_i ισχύει πάντα η ιδιότητα της διαμέρισης της μονάδας.

8 ΙΣΟΓΕΩΜΕΤΡΙΚΉ ΑΝΆΛΥΣΗ

Η βασική ιδέα πίσω από την Ισογεωμετρική Ανάλυση είναι ότι η βάση, με την οποία σχεδιάζεται με ακρίβεια ο γεωμετρικός χώρος εφαρμογής ενός προβλήματος συνοριακών τιμών, μπορεί να λειτουργήσει, επίσης, ως η βάση για το χώρο λύσεων της αριθμητικής μεθόδου επίλυσης του προβλήματος. Η ιδέα να χρησιμοποιήσουμε την ίδια βάση και για τη γεωμετρία και για την ανάλυση ονομάζεται *ισοπαραμετρική αρχή* και εφαρμόζεται και στην κλασική Ανάλυση Πεπερασμένων Στοιχείων (Finite Element Analysis, FEA), όπως αναφέρεται στο [2]. Επιπλέον, σημειώνεται ότι η θεμελιώδης διαφορά μεταξύ της Ισογεωμετρικής Ανάλυσης και της Ανάλυσης Πεπερασμένων Στοιχείων, είναι ότι σύμφωνα με τη δεύτερη προσέγγιση, η βάση, η οποία επιλέχθηκε για την προσέγγιση του *άγνωστου* πεδίου λύσεων είναι και η βάση που θα χρησιμοποιηθεί για την προσέγγιση της *γνωστής* γεωμετρίας. Η Ισογεωμετρική ανάλυση, όμως, αντιστρέφει αυτή τη διαδικασία κι έτσι, η βάση, η οποία επιλέγεται να περιγράψει την ακριβή γεωμετρία, χρησιμοποιείται για την προσέγγιση του άγνωστου πεδίου λύσεων. [2] Οι συναρτήσεις NURBS διαθέτουν σημαντικές ιδιότητες, οι οποίες κρίνονται χρήσιμες για την περιγραφή του πεδίου λύσεων, ενώ ένα υποσύνολο των συναρτήσεων T-spline κρίνεται εξίσου κατάλληλο για την εφαρμογή μεθόδων Ισογεωμετρικής Ανάλυσης.

Η ισοπαραμετρική αρχή μάς επιτρέπει να μπορούμε να διαχειριστούμε πολυπλοκότερες συναρτήσεις βάσης, από τις πολυωνυμικές της Ανάλυσης Πεπερασμένων Στοιχείων. Στο [2] αναφέρεται ότι για τη βάση της επιλογής μας θα πρέπει να ικανοποιούνται κάποιες προϋποθέσεις, οι οποίες, για αρχή, θα πρέπει να εξασφαλίζουν μια βασική σύγκλιση. Για ένα μεγάλο εύρος προβλημάτων, η σύγκλιση εξασφαλίζεται από μια βάση, η οποία είναι [20]:

- C^1 στο εσωτερικό των στοιχείων του χωρίου
- C^0 στο σύνορο των στοιχείων
- πλήρης

Ως γνωστόν, η προϋπόθεση της C^1-συνέχειας στο εσωτερικό των στοιχείων και η C^0-συνέχεια στο σύνορο των στοιχείων εξασφαλίζεται με άνεση από τις γεωμετρικές συναρτήσεις NURBS/T-spline. Τόσο οι συναρτήσεις NURBS όσο και οι συναρτήσεις βάσης T-spline είναι C^∞ στο εσωτερικό των στοιχείων και έχουν τουλάχιστον C^0-συνέχεια στο σύνορο των στοιχείων τους. Επιπρόσθετα, η τρίτη ιδιότητα εξασφαλίζεται από κάθε ισοπαραμετρική βάση που ικανοποιεί επιπλέον τη διαμέριση της μονάδας, σύμφωνα με το [2]. Επίσης, αναφέρεται ότι η ισοπαραμετρική αρχή και η διαμέριση της μονάδας είναι απαραίτητες για να διασφαλιστεί ότι η ισογεωμετρική ανάλυση θα οδηγήσει σε συγκλίνουσες μεθόδους, για πολλές διαφορετικές επιλογές συναρτήσεων.

Στην τεχνολογία των T-spline, χρειάζεται, πριν διερευνήσουμε την ιδιότητα της διαμέρισης της μονάδας, να μελετήσουμε κατά πόσο οι συναρτήσεις που κατασκευάζονται από μια συγκεκριμένη τοπολογία πλέγματος T-spline μπορούν να ορίσουν μια βάση για το χώρο T-spline. Αυτό το ερώτημα έχει απαντηθεί στο Κεφάλαιο 4 με τον ορισμό των «κατάλληλων-προς-ανάλυση T-spline» και στο Κεφάλαιο 7 με την περιγραφή κριτήριων που, αξιοποιώντας τον τελεστή εξαγωγής Bézier, επικυρώνουν την καθολική γραμμική ανεξαρτησία των συναρτήσεων T-spline, η οποία είναι αναγκαία και ικανή συνθήκη για τον ορισμό της βάσης από τις επιμέρους συναρτήσεις T-spline. Η περαιτέρω αξιοποίηση της βάσης T-spline στην Ισογεωμετρική Ανάλυση απαιτεί να ικανοποιείται και η ιδιότητα της πληρότητας, η οποία εξασφαλίζεται από τη διαμέριση της μονάδας. Επιπλέον, η

διαμέριση της μονάδας της βάσης συνεπάγεται τον αφινική συνδιακύμανση και την ικανοποίηση των patch tests. Σε αντίθεση με τις πολυδιάστατες συναρτήσεις NURBS και B-spline, που ορίζονται ως τανυστικά γινόμενα, η διαμέριση της μονάδας από την πολυωνυμική βάση T-spline δεν επαληθεύεται εκ των προτέρων και χρήζει διερεύνησης πριν προβούμε στη χρήση της βάσης σε αριθμητικές μεθόδους. Η διερεύνηση αυτή πραγματοποιείται με τη χρήση του τελεστή εξαγωγής Bézier, όπως συζητήθηκε στο Κεφάλαιο 7.1.2 ή εξετάζοντας αν το πλέγμα T-spline είναι αποδεκτό (Κεφάλαιο 4.4). Σε κάθε περίπτωση, είναι εφικτή η χρήση των ρητών συναρτήσεων T-spline ως βάση για την εφαρμογή της Ισογεωμετρικής Ανάλυσης, οι οποίες είναι γνωστό ότι ικανοποιούν πάντα τη διαμέριση της μονάδας.

8.1 Προβλήματα συνοριακών τιμών

Για την επεξήγηση της Ισογεωμετρικής Ανάλυσης στην επίλυση διαφορικών εξισώσεων, επιλέγεται ως παράδειγμα η εξίσωση Poisson, σύμφωνα με την προσέγγιση που ακολουθήθηκε στο [2].

Η εξίσωση Poisson είναι μια ελλειπτική διαφορική εξίσωση δευτέρου βαθμού και αποτελεί γενίκευση της εξίσωσης Laplace. Περιγράφει την κατανομή ενός βαθμωτού δυναμικού $u(x, y)$, π.χ. της θερμοκρασίας, σε ένα πεδίο Ω.

Το πρόβλημα Poisson διατυπώνεται ως εξής:

Να βρεθεί $u: \mathbb{R} \rightarrow \bar{\Omega}$, τέτοια ώστε

$$\nabla^2 u(x, y) = b(x, y) \text{ στο } \Omega \tag{8.1α}$$
$$u(x, y) = g \text{ στο } \Gamma_D \tag{8.1β}$$
$$\nabla u(x, y) \cdot \mathbf{n} = h \text{ στο } \Gamma_N \tag{8.1γ}$$
$$\beta u(x, y) + \nabla u(x, y) \cdot \mathbf{n} = r \text{ στο } \Gamma_R \tag{8.1δ}$$

όπου $\overline{\Gamma_D \cup \Gamma_N \cup \Gamma_R} = \Gamma \equiv \partial\Omega$, $\Gamma_D \cap \Gamma_N \cap \Gamma_R \neq \emptyset$ και με n συμβολίζεται το μοναδιαίο κάθετο διάνυσμα στο $\partial\Omega$. Οι συναρτήσεις $b: \Omega \rightarrow \mathbb{R}$, $g: \Gamma_D \rightarrow \mathbb{R}$, $h: \Gamma_N \rightarrow \mathbb{R}$ είναι γνωστές όπως και η σταθερά β. Η εξίσωση (8.1α) αποτελεί την *ισχυρή διατύπωση* του προβλήματος συνοριακών τιμών Poisson. Οι συνοριακές συνθήκες (8.1β), (8.1γ) και (8.1δ) αντιπροσωπεύουν τις τρεις μεγάλες κατηγορίες συνοριακών συνθήκων, που συναντάμε συχνότερα, δηλαδή, τις συνθήκες Dirichlet, Neumann και Robin, αντίστοιχα. Σε ένα επαρκώς ομαλό χωρίο και κάτω από συγκεκριμένες προϋποθέσεις, υπάρχει μοναδική λύση u που ικανοποιεί το πρόβλημα (8.1), αλλά πολύ συχνά είναι αδύνατο να εκφραστεί αναλυτικά. Για αυτό το λόγο χρησιμοποιούμε αριθμητικές μεθόδους για την προσέγγιση της ακριβούς λύσης. Στο πλαίσιο της Ισογεωμετρικής ανάλυσης, και ανάλογα με το είδος της παρεμβολής που επιλέγουμε, η λύση u_h, η οποία προσεγγίζει την ακριβή λύση u γράφεται σε μία από τις ακόλουθες εκδοχές:

B-spline
$$u^h(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(\xi) N_{j,q}(\eta) \cdot a_{ij} \tag{8.2}$$

NURBS
$$u^h(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,p}(\xi) R_{j,q}(\eta) \cdot a_{ij} \tag{8.3}$$

T-spline $$u^h(x,y) = \sum_{a=1}^{N} R_a(\xi,\eta) \cdot a_a \tag{8.4}$$

Πιο συγκεκριμένα, στις περιπτώσεις παρεμβολής B-spline και NURBS, υπονοείται καρτεσιανό επίπεδο με $n+1$ σημεία ελέγχου στην κατεύθυνση x και $m+1$ σημεία ελέγχου στην κατεύθυνση y. Έτσι, συνολικά, χρησιμοποιούνται $(n+1)\cdot(m+1)$ σημεία ελέγχου και συναρτήσεις βάσης. Αντίθετα, στην περίπτωση των T-spline, χρησιμοποιείται ένας δείκτης, καθώς το πλέγμα ελέγχου δεν συνιστά καρτεσιανό πλέγμα. Επιπλέον, σημειώνεται ότι στην περίπτωση των T-spline με σταθερό συντελεστή βάρους ανά δεσμό, w_a, οι ρητές συναρτήσεις R_a ταυτίζονται με τις αντίστοιχες πολυωνυμικές συναρτήσεις N_a.

Σε κάθε περίπτωση, σκοπός είναι μέσα από την εκάστοτε αριθμητική μέθοδο να υπολογιστούν οι συντελεστές a_{ij} ή a_a, έτσι ώστε $u^h \approx u$. Για την επίτευξη του εν λόγω σκοπού, ακολουθείται η μέθοδος Bubnov-Galerkin.

8.1.1 Μέθοδος Galerkin

Για την επίλυση του πρόβλημα συνοριακών τιμών Poisson, θα πρέπει να διατυπωθεί στην *ασθενή μορφή* του (weak form). Δε χρειάζεται η προσεγγιστική λύση u^h να ικανοποιεί τη διαφορική εξίσωση Poisson στην αρχική της μορφή, αλλά μόνο την ασθενή εκδοχή της. Επιπλέον, η ασθενής μορφή λειτουργεί κι ως σημείο αφετηρίας για τη διαδικασία της διακριτοποίησης.

8.1.1.1 Ασθενής μορφή του προβλήματος Poisson

Για τον σχηματισμό της ασθενούς μορφής του προβλήματος συνοριακών τιμών, ορίζονται δυο συλλογές συναρτήσεων. Η πρώτη συλλογή, η συλλογή των *συναρτήσεων δοκιμής* (trial functions) συμβολίζεται με $\mathcal{S}$ και περιέχει όλες τις συναρτήσεις οι οποίες έχουν τετραγωνικά ολοκληρώσιμες παραγώγους και για τις οποίες ισχύει

$$u|_{\Gamma_D} = g. \tag{8.5}$$

Η συλλογή των συναρτήσεων δοκιμής γράφεται, επομένως, ως

$$\mathcal{S} = \{u | u \in H^1(\Omega), u|_{\Gamma_D} = g\}. \tag{8.6}$$

Η δεύτερη συλλογή συναρτήσεων περιέχει τις *συναρτήσεις στάθμισης* (weighting functions) και είναι παρόμοια με τη συλλογή των συναρτήσεων δοκιμής με τη διαφορά ότι στο σύνορο Dirichlet επιβάλλεται η ομογενής συνοριακή συνθήκη Dirichlet. Το σύνολο των συναρτήσεων στάθμισης συμβολίζεται με $\mathcal{V}$ και ορίζεται ως

$$\mathcal{V} = \{w \,|w \in H^1(\Omega), w|_{\Gamma_D} = 0\}. \tag{8.7}$$

Για τον υπολογισμό της ασθενούς διατύπωσης του προβλήματος συνοριακών τιμών, η εξίσωση (8.1α) πολλαπλασιάζεται με μια αυθαίρετη συνάρτηση ελέγχου (test function) $w \in \mathcal{V}$. Έπειτα, ολοκληρώνοντας κατά μέρη και ενσωματώνοντας τις σχέσεις (8.1γ) και (8.1δ), λαμβάνουμε την ασθενή διατύπωση του προβλήματος, η οποία είναι:

$$\int_{\Omega} \nabla w \cdot \nabla u \, d\Omega + \beta \int_{\Gamma_R} wu \, d\Gamma = \int_{\Omega} wf \, d\Omega + \int_{\Gamma_N} wh \, d\Gamma + \int_{\Gamma_R} wr \, d\Gamma . \tag{8.8}$$

Επισημαίνεται ότι όλη η άγνωστη πληροφορία, δηλαδή η συνάρτηση u, περιέχεται στην αριστερή πλευρά της εξίσωσης, ενώ όλα τα γνωστά δεδομένα, οι συναρτήσεις h και r, περιέχονται στη δεξιά πλευρά.

Υπενθυμίζεται ότι ο χώρος Sobolev $H^1(\Omega)$ ορίζεται ως

$$H^1(\Omega) = \{u | D^{\boldsymbol{\alpha}} u \in L^2(\Omega), |\boldsymbol{\alpha}| \leq 1\} \tag{8.9}$$

όπου $\boldsymbol{\alpha} \in \mathbb{N}^d$ είναι ένα διάνυσμα όπου d η διάσταση του χωρίου Ω, $|\boldsymbol{\alpha}| = \sum_{i=1}^{d} a_i$ και $D^a = D_1^{a_1} D_2^{a_2} \dots D_d^{a_d}$, όπου $D_i^j = \frac{\partial^j}{\partial x_i^j}$ ο διαφορικός τελεστής. Επίσης, ο χώρος $L^2(\Omega)$, που εμφανίζεται στον ορισμό του χώρου $H^1(\Omega)$ ορίζεται ως το σύνολο όλων των τετραγωνικά ολοκληρώσιμων συναρτήσεων, που ορίζονται στο χωρίο $\Omega \subset \mathbb{R}^d$, δηλαδή, ως η συλλογή όλων των συναρτήσεων $u: \Omega \to \mathbb{R}$ για τις οποίες ισχύει

$$\int_{\Omega} u^2 d\Omega < +\infty \tag{8.10}$$

Από την ασθενή διατύπωση του προβλήματος γίνεται σαφές για ποιο λόγο επιλέγεται ο χώρος $H^1(\Omega)$ για τον ορισμό των συναρτήσεων δοκιμής και στάθμισης. Παρά το γεγονός ότι η ισχυρή διατύπωση της εξίσωσης (8.1α) απαιτεί η u να έχει καλώς ορισμένες δεύτερες παραγώγους, η ασθενής διατύπωση απαιτεί απλώς οι πρώτες παράγωγοι να είναι τετραγωνικά ολοκληρώσιμες, δηλαδή η συνάρτηση u να ανήκει στο χώρο $H^1(\Omega)$.

Η ασθενή μορφή μπορεί να επαναδιατυπωθεί ως

$$a(w, u) = L(w) \tag{8.11}$$

όπου

$$a(w, u) = \int_{\Omega} \nabla w \cdot \nabla u \, d\Omega + \beta \int_{\Gamma_R} wu \, d\Gamma , \tag{8.12}$$

και

$$L(w) = \int_{\Omega} wf \, d\Omega + \int_{\Gamma_N} wh \, d\Gamma + \int_{\Gamma_R} wr \, d\Gamma . \tag{8.13}$$

Τα συναρτησιακά $a(\cdot,\cdot)$ και $L(\cdot)$ εμφανίζουν μερικές ενδιαφέρουσες ιδιότητες, όπως η συμμετρία και η διγραμμικότητα για το $a(\cdot,\cdot)$ και η γραμμικότητα για το $L(\cdot)$.

Η λύση της εξίσωσης (8.8), ή αντίστοιχα της (8.11), καλείται *ασθενής λύση*. Κάτω από κατάλληλες παραδοχές, αποδεικνύεται ότι η ασθενής και η ισχυρή λύση της (8.1α) είναι ισοδύναμες [20].

8.1.1.2 Διακριτοποίηση Galerkin

Στη μέθοδο Galerkin κατασκευάζονται πεπερασμένης διάστασης προσεγγίσεις των χώρων $\mathcal{S}$ και $\mathcal{V}$. Στο πλαίσιο της ισογεωμετρικής ανάλυσης, οι προσεγγίσεις αυτές συνιστούν τους υπο-χώρους $\mathcal{S}^h \subset \mathcal{S}$ και $\mathcal{V}^h \subset \mathcal{V}$, οι οποίοι παράγονται από τις ίδιες συναρτήσεις βάσης που κατασκευάζουν το φυσικό χωρίο.

Μπορούμε να χαρακτηρίσουμε περαιτέρω τον υπόχωρο $\mathcal{S}^h$, παρατηρώντας ότι αν θεωρήσουμε μια $g^h \in \mathcal{S}^h$, για την οποία ισχύει $g^h\big|_{\Gamma_D} = g$, τότε για κάθε $u^h \in \mathcal{S}^h$ υπάρχει μοναδική $v^h \in \mathcal{V}^h$ τέτοια ώστε

$$u^h = v^h + g^h. \tag{8.14}$$

Η παραπάνω σχέση δεν επαληθεύεται για οποιαδήποτε συνάρτηση g, παρ' όλα αυτά για λόγους απλότητας, λαμβάνουμε ως δεδομένο ότι ισχύει και υπάρχει μια g^h.

Κατά αυτόν τον τρόπο, η μεταβολική διατύπωση Galerkin του προβλήματος συνοριακών τιμών (8.1α) ορίζεται ως εξής: Δεδομένων των g^h, h και r, να βρεθεί $u^h = v^h + g^h$, όπου $v^h \in \mathcal{V}^h$, έτσι ώστε για όλα τα w^h στο χώρο $\mathcal{V}^h$ να ισχύει

$$a\left(w^h, u^h\right) = L\left(w^h\right). \tag{8.15}$$

Λόγω της (3.22) και της διγραμμικότητας του $a(\cdot,\cdot)$, η εξίσωση (3.23) διατυπώνεται ως εξής

$$a\left(w^h, u^h\right) = L\left(w^h\right) - a\left(w^h, g^h\right). \tag{8.16}$$

Έτσι, η άγνωστη πληροφορία είναι στην αριστερή πλευρά και στα δεξιά βρίσκονται όλα τα δεδομένα. Οι εξισώσεις (8.15) και (8.16) αναφέρονται και ως η μέθοδος Bubnov-Galerkin.

Στην Ισογεωμετρική Ανάλυση, χρησιμοποιείται η ισοπαραμετρική αρχή, δηλαδή ο χώρος λύσεων αναπαρίσταται ως το γραμμικό ανάπτυγμα των γεωμετρικών συναρτήσεων βάσης NURBS/T-spline $R_A: \widehat{\Omega} \to \mathbb{R}$, όπου $A = 1, \dots, n$, με n το πλήθος των συναρτήσεων. Καθώς, οι συναρτήσεις βάσης έχουν ιδιαίτερα τοπικό φορέα, λίγες συναρτήσεις θα είναι μη μηδενικές στο σύνορο Dirichlet. Επομένως, μπορούμε να υποθέσουμε χωρίς απώλεια της γενικότητας ότι υπάρχουν $n_{eq} < n$ συναρτήσεις βάσης για τις οποίες ισχύει

$$R_A|_{\Gamma_D} = 0 \; \forall A = 1, \dots, n_{eq}. \tag{8.17}$$

Ως εκ τούτου, για κάθε $w^h \in \mathcal{V}^h$ υπάρχουν σταθερές c_A τέτοιες ώστε

$$w^h = \sum_{A=1}^{n_{eq}} R_A c_A \,. \tag{8.18}$$

Επιπλέον, η συνάρτηση $g^h \in \mathcal{S}^h$ μπορεί να γραφεί σε ανάλογη μορφή με τη βοήθεια συντελεστών g_A, $A = 1, \dots, n$, μιας και ο χώρος $\mathcal{S}^h$ είναι γραμμικό ανάπτυγμα των συναρτήσεων βάσης. Στην

πράξη, επιλέγεται συνάρτηση g^h τέτοια ώστε $g_1 = \cdots = g_{n_{eq}} = 0$ ώστε οι συντελεστές να μην επηρεάζουν την τιμή της συνάρτησης στο σύνορο Γ_D. Έτσι έχουμε

$$g^h = \sum_{A=n_{eq}+1}^{n} R_A g_A \,. \tag{8.19}$$

Τέλος, ανακαλώντας την (8.14), έχουμε ότι για κάθε $u^h \in \mathcal{S}^h$, υπάρχουν d_A, $A = 1, \dots, n_{eq}$ τέτοια ώστε

$$u^h = \sum_{A=1}^{n_{eq}} R_A d_A + \sum_{B=n_{eq}+1}^{n} R_B g_B = \sum_{A=1}^{n_{eq}} R_A d_A + g^h. \tag{8.20}$$

Εισάγοντας τις εξισώσεις (8.18) και (8.20) στην (8.16) και αξιοποιώντας τη γραμμικότητα του ολοκληρώματος καθώς και το γεγονός ότι οι μεταβλητές c_A είναι αυθαίρετες, καταλήγουμε στην εξίσωση

$$\sum_{B=1}^{n_{eq}} a(R_A, R_B) d_B = L(R_A) - a\big(R_A, g^h\big) \tag{8.21}$$

για $A = 1, \dots n_{eq}$.

Αντικαθιστώντας τις παραπάνω σχέσεις στην εξίσωση Galerkin και διατυπώνοντας το σύστημα σε μητρωική μορφή, παίρνουμε

$$\mathbf{Kd} = \mathbf{F}, \tag{8.22}$$

όπου

$$\mathbf{K} = [K_{AB}], \tag{8.23}$$
$$\mathbf{F} = \{F_A\}, \tag{8.24}$$
$$\mathbf{d} = \{d_B\}, \tag{8.25}$$

$$K_{AB} = a(R_A, R_B) \tag{8.26}$$
$$F_A = L(R_A) - a\big(R_A, g^h\big) \tag{8.27}$$

Το μητρώο $\mathbf{K} \in \mathbb{R}^{n \times n}$ καλείται μητρώο στιβαρότητας, το διάνυσμα $\mathbf{F} \in \mathbb{R}^{n \times 1}$ διάνυσμα δυνάμεων και το $\mathbf{d} \in \mathbb{R}^{n \times 1}$ διάνυσμα μετατοπίσεων.

Λύνοντας την εξίσωση (8.22) ως προς τα d_A για $A = 1, \dots n$, έχουμε

$$\mathbf{d} = \mathbf{K}^{-1}\mathbf{F} \tag{8.28}$$

Και εισάγοντας τα d_A στην εξίσωση (8.20),η τελική λύση u^h γράφεται ως

$$u^h = \sum_{A=1}^{n_{eq}} R_A d_A + \sum_{B=n_{eq}+1}^{n} R_B g_B \,. \tag{8.29}$$

Η προαναφερθείσα διατύπωση εφαρμόζεται μόνο σε μερικές διαφορικές εξισώσεις βαθμωτού πεδίου, όπως η εξίσωση μεταφοράς θερμότητας. Για την επίλυση διανυσματικών μερικών διαφορικών εξισώσεις, όπως στα προβλήματα ελαστικότητας ακολουθείται διαφορετική μεθοδολογία, η οποία περιγράφεται στα [2, 20].

8.1.1.3 Συναρμολόγηση συστήματος

Το μητρώο στιβαρότητας **K** είναι ένα αραιό μητρώο, απόρροια του ιδιαίτερα τοπικού φορέα κάθε συνάρτησης βάσης. Έτσι, για πολλούς συνδυασμούς δεικτών A και B στο $n \times n$ ολικό μητρώο στιβαρότητας θα ισχύει $K_{AB} = a(N_A, N_B) = 0$. Αξιοποιώντας αυτό το γεγονός, μπορεί να μειωθεί ο υπολογιστικός χρόνος για την κατασκευή και επίλυση του αλγεβρικού συστήματος. Σε κάθε στοιχείο NURBS/T-spline υπάρχει ένα καθορισμένο πλήθος μη μηδενικών συναρτήσεων βάσης. Στην περίπτωση της βάσης NURBS, το πλήθος αυτό είναι σταθερό για κάθε στοιχείο και ίσο με $(p+1) \cdot (q+1)$. Αντίθετα στη βάση T-spline, το πλήθος των μη μηδενικών συναρτήσεων διαφέρει σε κάθε στοιχείο, λόγω των διασταυρώσεων T, αν και σε κάθε περίπτωση δεν είναι μικρότερο από $(p+1) \cdot (q+1)$. Έστω $n_{en}(e)$ το πλήθος των τοπικών συναρτήσεων για το στοιχείο e, δηλαδή το πλήθος των μη μηδενικών συναρτήσεων του στοιχείου. Το τοπικό ή στοιχειώδες μητρώο στιβαρότητας $\mathbf{k}^e$ για ένα στοιχείο e, διάστασης $n_{en}(e) \times n_e(e)$ θα είναι ένα πυκνό μητρώο, καθώς κατασκευάζεται από τις μη μηδενικές συναρτήσεις του στοιχείου. Ο σχηματισμός του τελικού ολικού μητρώου στιβαρότητας **K** συνίσταται στον υπολογισμό των τοπικών μητρώων στιβαρότητας $\mathbf{k}^e$ για όλα τα στοιχεία e και στην τοποθέτηση τους στις κατάλληλες θέσεις του μητρώου **K**. Με αυτόν τον τρόπο, δεν χρειάζεται να καταβάλλουμε προσπάθεια για την ολοκλήρωση συναρτήσεων σε περιοχές του στις οποίες γνωρίζουμε εκ των προτέρων ότι οι συναρτήσεις μηδενίζονται.

Η παραπάνω διαδικασία απλοποιείται με έναν πίνακα συνδεσιμότητας, ο οποίος καλείται IEN και συνδέει κάθε αριθμό τοπικής συνάρτησης βάσης με έναν αριθμό καθολικής συνάρτησης βάσης. Για κάθε αριθμό στοιχείου e, με $e = 1, \dots, n_{el}$ και κάθε αριθμό τοπικής συνάρτησης βάσης a, με $a = 1, \dots, n_{en}(e)$, υπάρχει μια καθολική συνάρτηση βάσης με αριθμό $A = 1, \dots, n$, έτσι ώστε $\mathrm{IEN}(a, e) = A$. Δηλαδή, η τοπική συνάρτηση N_a του στοιχείου e ταυτίζεται με την καθολική συνάρτηση N_A. Έτσι, το καθολικό μητρώο στιβαρότητας μπορεί να κατασκευαστεί από μια αλληλουχία τοπικών μητρώων. Με παρόμοιο τρόπο το καθολικό διάνυσμα δυνάμεων **F** συναρμολογείται από τα τοπικά διανύσματα δυνάμεων $\mathbf{F}^e$.

Κατά τη συναρμολόγηση των καθολικών μεγεθών, θα πραγματοποιείται ολοκλήρωση μόνο των συναρτήσεων που είναι μη μηδενικές, σύμφωνα με τον κανόνα ολοκλήρωσης Gauss. Όπως φαίνεται και στο Σχήμα 8-1 τα ολοκληρώματα, τα οποία ορίζονται στο φυσικό στοιχείο Ω^e επιστρέφουν μέσω της αντίστροφης γεωμετρικής απεικόνισης $\mathbf{x}^{-1}$ στο παραμετρικό στοιχείο $\widehat{\Omega}^e$ του παραμετρικού χωρίου κι έπειτα μέσω της αντίστροφης αφινικής απεικόνισης $\boldsymbol{\varphi}^{-1}$ στο γονικό στοιχείο $\widetilde{\Omega}^e = [-1,1]^2$, όπου πραγματοποιείται η ολοκλήρωση Gauss, με μια κατάλληλη τροποποίηση των μεταβλητών ολοκλήρωσης. Θεωρούμε ότι οι συντεταγμένες στο φυσικό χωρίο συμβολίζονται ως $\mathbf{x}$, οι συντεταγμένες στο παραμετρικό χωρίο ως $\boldsymbol{\xi}$ και στο γονικό στοιχείο ως $\tilde{\boldsymbol{\xi}}$. Χρησιμοποιώντας την αντιστοίχιση της καθολικής αρίθμησης των συναρτήσεων, $A = 1, \dots, n$ με την τοπική αρίθμηση $a = 1, \dots, n_{en}(e)$ για κάθε στοιχείο $\widehat{\Omega}^e$, $e = 1, \dots, n_{el}$, σύμφωνα με τη σχέση $A = \mathrm{IEN}(a, e)$, η απεικόνιση $\mathbf{x}^e : \widehat{\Omega}^e \to \Omega^e$ που αντιστοιχίζει κάθε παραμετρικό στοιχείο σε ένα φυσικό στοιχείο ορίζεται ως

$$\mathbf{x}^e(\boldsymbol{\xi}) \equiv \begin{Bmatrix} x^e(\boldsymbol{\xi}) \\ y^e(\boldsymbol{\xi}) \\ z^e(\boldsymbol{\xi}) \end{Bmatrix} = \frac{\sum_{a=1}^{n_{en}} w_a^e N_a^e(\boldsymbol{\xi}) \mathbf{P}_a^e}{W^e(\boldsymbol{\xi})}$$

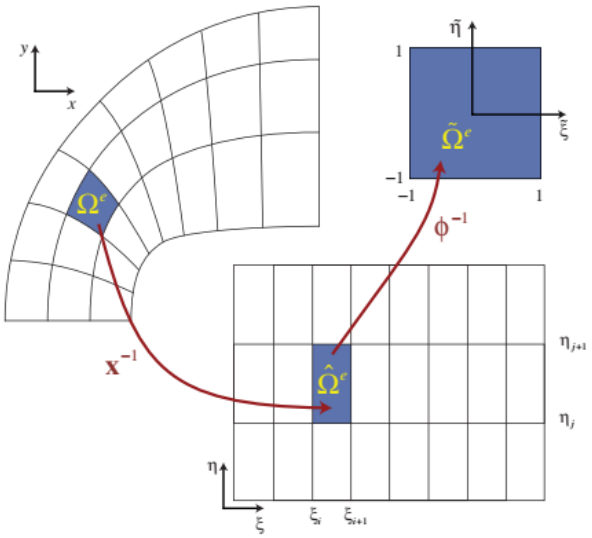
$$= \sum_{a=1}^{n_{en}} R_a^e(\boldsymbol{\xi}) \mathbf{P}_a^e \tag{8.30}$$

όπου

$$W^e(\boldsymbol{\xi}) = \sum_{a=1}^{n_{en}} w_a^e N_a^e(\boldsymbol{\xi}) \tag{8.31}$$

η στοιχειώδης συνάρτηση βάρους, $\mathbf{P}_a^e$ το σημείο ελέγχου, N_a^e η συνάρτηση βάσης και w_a^e ο συντελεστής βάρους, που αντιστοιχούν στο στοιχείο $\widehat{\Omega}^e$, σύμφωνα με το μητρώο IEN. Επιπλέον, με R_a^e συμβολίζεται η ρητή μορφή της σύναρτησης βάσης (NURBS ή T-spline).

Η επαναφορά από το φυσικό χωρίο στο γονικό στοιχείο, για την υλοποίηση της ολοκλήρωσης Gauss, πραγματοποιείται με τη σύνθεση των αντίστροφων των δυο παραπάνω απεικονίσεων, όπως στο Σχήμα 8-1. Σε κάθε σημείο Gauss του γονικού στοιχείου, υπολογίζονται οι συναρτήσεις βάσης, οι κλίσεις τους και η ιακωβιανή ορίζουσα της σύνθεσης. Όλες οι παραπάνω διαδικασίες πραγματοποιούνται στη ρουτίνα συνάρτησης μορφής (shape function routine), η οποία, μάλιστα, διαφοροποιείται ανάλογα με τη μέθοδο Ισογεωμετρικής Ανάλυσης που επιλέγεται. Το συγκεκριμένο ζήτημα θα μελετηθεί αναλυτικότερα στο Κεφάλαιο 8. Αξίζει να τονιστεί ότι ενώ οι συναρτήσεις NURBS όσο και οι ρητές συναρτήσεις T-spline δεν είναι κατά κανόνα πολυώνυμα, ο κανόνας Gauss φαίνεται να είναι αρκετά αποτελεσματικός για την ολοκλήρωση τους. Μάλιστα, μπορεί να χρησιμοποιηθεί για μια ρητή συνάρτηση NURBS/T-spline p βαθμού ο κανόνας Gauss που θα χρησιμοποιούσαμε για ένα πολυώνυμο βαθμού p.



Σχήμα 8-1 Η ολοκλήρωση με τον κανόνα Gauss πραγματοποιείται πάντα στο γονικό στοιχείο. Το φυσικό στοιχείο επαναφέρεται αρχικά στο παραμετρικό χωρίο μέσω της αντίστροφης γεωμετρικής απεικόνισης κι έπειτα μέσω μιας αφινικής απεικόνισης στο γονικό στοιχείο, με τις κατάλληλες μετατροπές των μεταβλητών. [2]

8.2 Γεωμετρική απεικόνιση

Περιορίζοντας τις γεωμετρικές συναρτήσεις βάσης στο γονικό στοιχείο $\widetilde{\Omega}^e$, προκύπτει μια οπτική που προσομοιάζει με την προσέγγιση της Ανάλυσης Πεπερασμένων Στοιχείων. Σύμφωνα με τη σχέση $A = \mathrm{IEN}(a, e)$, κάθε καθολική συνάρτηση βάσης γράφεται ως

$$N_A(\boldsymbol{\xi})|_e = N_A\left(\boldsymbol{\varphi}(\tilde{\boldsymbol{\xi}})\right)\Big|_e = N_a^e(\tilde{\boldsymbol{\xi}}). \tag{8.32}$$

όπου $\boldsymbol{\varphi}: \widetilde{\Omega}^e \to \widehat{\Omega}^e$, η αφινική απεικόνιση από το γονικό στοιχείο προς το στοιχείο στο παραμετρικό χωρίο.

Επομένως, μπορεί να οριστεί γεωμετρική απεικόνιση $\tilde{\mathbf{x}}^e: \widetilde{\Omega}^e \to \Omega^e$, από το γονικό στοιχείο κατευθείαν στο φυσικό στοιχείο ως

$$\tilde{\mathbf{x}}^e(\boldsymbol{\xi}) = \frac{\sum_{a=1}^{n_{en}} w_a^e N_a^e(\tilde{\boldsymbol{\xi}})\mathbf{P}_a^e}{W^e(\tilde{\boldsymbol{\xi}})} = \sum_{a=1}^{n_{en}} R_a^e(\tilde{\boldsymbol{\xi}})\mathbf{P}_a^e \tag{8.33}$$

όπου

$$W^e(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_{en}} w_a^e N_a^e(\tilde{\boldsymbol{\xi}}) \tag{8.34}$$

Η μητρωική εκδοχή της σχέσης (8.33) είναι

$$\begin{aligned}\tilde{\mathbf{x}}^e(\tilde{\xi}) &= \frac{1}{(\mathbf{w}^e)^T\mathbf{N}^e(\tilde{\boldsymbol{\xi}})}(\mathbf{P}^e)^T\mathbf{W}^e\mathbf{N}^e(\tilde{\boldsymbol{\xi}}) \\ &= (\mathbf{P}^e)^T\mathbf{R}^e(\tilde{\boldsymbol{\xi}}),\end{aligned} \tag{8.35}$$

όπου $\mathbf{N}^e = \{N_a^e\}_{a=1}^{n_{en}}$ και $\mathbf{R}^e = \{R_a^e\}_{a=1}^{n_{en}}$ είναι τα διανύσματα των πολυωνυμικών και ρητών συναρτήσεων, αντίστοιχα, οι οποίες είναι μη μηδενικές στο στοιχείο Ω^e.

Η εν λόγω γεωμετρική απεικόνιση από το γονικό στο φυσικό στοιχείο θα φανεί ιδιαίτερα χρήσιμη στην Αποσύνθεση Bézier (Κεφάλαιο 6) και στις εφαρμογές της στην Ισογεωμετρική Ανάλυση (Κεφάλαιο 9.2, Κεφάλαιο 9.3).

9 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΣΥΝΟΡΙΑΚΩΝ ΤΙΜΩΝ ΣΤΗΝ ΙΣΟΓΕΩΜΕΤΡΙΚΗ ΑΝΑΛΥΣΗ

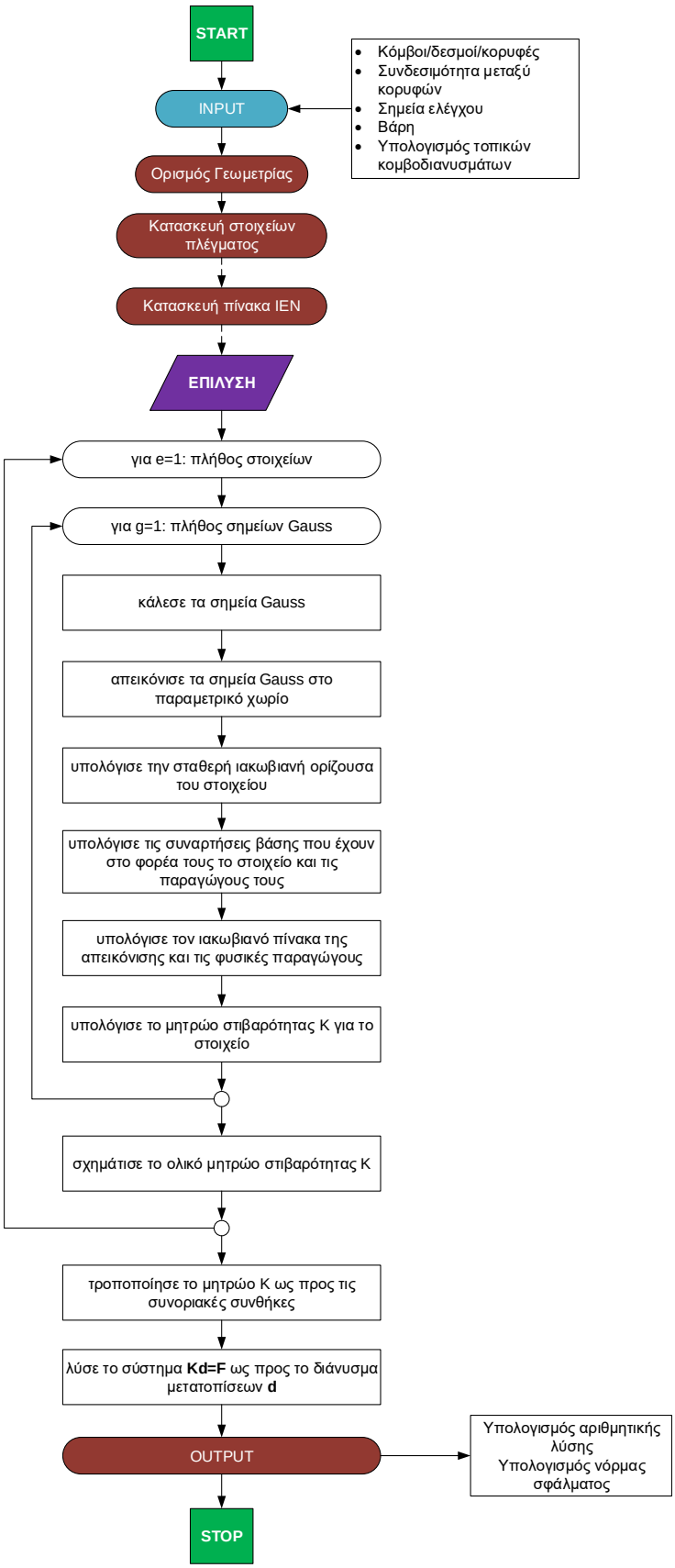
Η μοντελοποίηση ενός προβλήματος συνοριακών τιμών χωρίζεται σε δύο μέρη. Το πρώτο μέρος σχετίζεται με την κατασκευή της γεωμετρίας του προβλήματος και περιγράφηκε στο Κεφάλαιο 8. Αφορά τον ορισμό των παραμετρικών συντεταγμένων των δεσμών του πλέγματος, της τοπολογικής σχέσης μεταξύ τους και των φυσικών συντεταγμένων των σημείων ελέγχου. Σε αυτό το σημείο, έχουμε όλα τα απαιτούμενα δεδομένα για τη σχεδίαση της επιφάνειας T-spline.

Το δεύτερο μέρος αφορά την διατύπωση και επίλυση του προβλήματος συνοριακών τιμών που ορίζεται πάνω στην προαναφερθείσα γεωμετρία.

9.1 Υπολογιστικές διαδικασίες στη συμβατική Ισογεωμετρική Ανάλυση

Στη συμβατική Ισογεωμετρική Ανάλυση, το μπλοκ του κώδικα περιλαμβάνει τους εξής υπολογισμούς:

1. Σχηματίζεται το στοιχειώδες πλέγμα T-spline (elemental T-mesh) και τα επιμέρους στοιχεία (elements) που το απαρτίζουν. Όπως έχει ήδη αναφερθεί, τα στοιχεία είναι οι υπο-περιοχές του πλέγματος στα οποία πραγματοποιείται η αριθμητική ολοκλήρωση Gauss.
2. Πριν από την συναρμολόγηση, προτείνεται η κατασκευή του πίνακα IEN. Ο πίνακας IEN αντιστοιχίζει σε κάθε στοιχείο τις συναρτήσεις βάσης που είναι μη μηδενικές σε αυτό. Η χρήση του IEN μειώνει το υπολογιστικό κόστος, καθώς για κάθε στοιχείο λαμβάνονται υπόψιν μόνο οι συναρτήσεις που έχουν φορέα στο στοιχείο και όχι όλες οι συναρτήσεις του T-spline. Σε προβλήματα με μεγάλο πλήθος βαθμών ελευθερίας, η εξοικονόμηση αυτή αντικατοπτρίζεται και στο συνολικό υπολογιστικό χρόνο.
3. Συναρμολογείται το μητρώο στιβαρότητας. Κατά τη συναρμολόγηση του μητρώου, ακολουθούνται τα παρακάτω βήματα σε κάθε στοιχείο του T-spline:
   - Καλούνται τα σημεία Gauss του γονικού στοιχείου και απεικονίζονται στον παραμετρικό χώρο.
   - Σε κάθε μετασχηματισμένο σημείο Gauss υπολογίζονται οι συναρτήσεις βάσης και οι μερικές παράγωγοι τους στο παραμετρικό χωρίο.
   - Υπολογίζεται η ορίζουσα της απεικόνισης από το γονικό στοιχείο $[-1,1]^2$ στο παραμετρικό στοιχείο. Καθώς κάθε παραμετρικό στοιχείο είναι ορθογώνιο τετράπλευρο με κορυφές $ABCD$, η ορίζουσα είναι ένας σταθερός, για κάθε στοιχείο, αριθμός $\frac{|AB|}{2} \cdot \frac{|AD|}{2}$.
   - Υπολογίζονται ο ιακωβιανός πίνακας και η ορίζουσα της απεικόνισης του στοιχείου από το παραμετρικό στο φυσικό χωρίο καθώς και οι μερικές παράγωγοι ως προς τις φυσικές συντεταγμένες στα σημεία Gauss του γονικού στοιχείου.
4. Στο σύστημα εξισώσεων του προβλήματος επιβάλλονται οι συνθήκες Dirichlet, μετασχηματίζοντας το μητρώο στιβαρότητας. Τέλος, το σύστημα στην τελική του εκδοχή επιλύεται ως προς τις μεταβλητές ελέγχου της αριθμητικής λύσης.
5. Η αριθμητική λύση του προβλήματος υπολογίζεται ως γραμμικός συνδυασμός των συναρτήσεων βάσης με συντελεστές τις μεταβλητές ελέγχου.



Σχήμα 9-1Αλγόριθμος υπολογισμού της αριθμητικής λύσης σύμφωνα με τη συμβατική Ισογεωμετρική Ανάλυση

9.1.1 Ρουτίνα συνάρτησης μορφής

Η ρουτίνα συνάρτησης μορφής (ή βάσης) είναι αναπόσπαστο κομμάτι ενός κώδικα πεπερασμένων στοιχείων. Δεδομένου ενός αριθμού στοιχείου, e, και σημείων ολοκλήρωσης στο γονικό στοιχείο $(\tilde{\xi}_1, \tilde{\xi}_2) = (\tilde{\xi}, \tilde{\eta}) \in [-1,1]^2$, η ρουτίνα υπολογίζει κάθε μια από τις τοπικές συναρτήσεις βάσης στο στα σημεία ολοκλήρωσης, καθώς και τις απαιτούμενες φυσικές παραγώγους. Επιπλέον, για την πραγματοποίηση της ολοκλήρωσης, υπολογίζεται και η ιακωβιανή ορίζουσα της απεικόνισης.

Η ρουτίνα συνάρτησης μορφής προϋποθέτει την ύπαρξη κι άλλων ρουτινών, οι οποίες θα υπολογίζουν τις μονοδιάστατες συναρτήσεις B-spline και τις παραμετρικές παραγώγους τους. Καθώς, μας ενδιαφέρει η επίλυση προβλημάτων με συναρτήσεις T-spline, η ρουτίνα υπολογισμού των συναρτήσεων T-spline θα πρέπει να λαμβάνει ως μεταβλητή εισόδου το τοπικό κομβοδιάνυσμα της συνάρτησης και το σημείο υπολογισμού. Στο πλαίσιο του υπολογιστικού πακέτου, στο οποίο βασίζεται η παρούσα εργασία, η ρουτίνα αυτή υλοποιείται από τη συνάρτηση `basisFunTSpl`[2]. Έστω, για παράδειγμα, ένα στοιχείο $\widehat{\Omega}^{\mathrm{e}} = [\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$. Στην κατεύθυνση ξ, η συνάρτηση `basisFunTSpl` θα επιστρέψει την τιμή της συνάρτησης T-spline, η οποία ορίζεται για το δεδομένο τοπικό κομβοδιάνυσμα. Επίσης, η ρουτίνα `derRecur` εκτυπώνει τις παραμετρικές παραγώγους της εν λόγω συνάρτησης T-spline. Εκτελώντας τις παραπάνω συναρτήσεις για το κομβοδιάνυσμα κάθε συνάρτησης, η οποία στηρίζεται στο στοιχείο $\widetilde{\Omega}^e$, λαμβάνουμε τις τιμές των τουλάχιστον $(p+1) \cdot (q+1)$ συναρτήσεων και των παραγώγων τους. Με αυτές τις τιμές των μονοδιάστατων, μη ρητών συναρτήσεων, οι διδιάστατες, ρητές συναρτήσεις NURBS/T-spline, R, υπολογίζονται χρησιμοποιώντας την (2.30). Οι παράγωγοι ως προς τις παραμετρικές συντεταγμένες,$\boldsymbol{\xi}$, υπολογίζονται από τη σχέση (2.31). Για τον υπολογισμό των παραγώγων ως προς τις φυσικές συντεταγμένες $(x_1, x_2) \equiv (x, y)$, εφαρμόζεται ο κανόνας της αλυσίδας

$$\frac{\partial R}{\partial x_i} = \sum_{j=1}^{2} \frac{\partial R}{\partial \xi_j} \frac{\partial \xi_j}{\partial x_i} \,. \tag{9.1}$$

Επομένως, θα πρέπει να υπολογιστεί προηγουμένως η κλίση της απεικόνισης, $\partial \mathbf{x}/\partial \boldsymbol{\xi}$, καθώς και η αντίστροφη της. Ο υπολογισμός της αντίστροφης απεικόνισης υλοποιείται, εφαρμόζοντας τον κανόνα Cramer.

Τέλος, καθώς η ρουτίνα συνάρτησης μορφής εκτελεί την αριθμητική ολοκλήρωση στο γονικό στοιχείο, πρέπει να υπολογιστεί η ιακωβιανή ορίζουσα της απεικόνισης από το γονικό στοιχείο στο φυσικό χωρίο, $\det\mathbf{J}$.

$$\det\mathbf{J} = \left|\frac{d\mathbf{x}}{d\tilde{\boldsymbol{\xi}}}\right| = \left|\frac{d\mathbf{x}}{d\boldsymbol{\xi}} \frac{d\boldsymbol{\xi}}{d\tilde{\boldsymbol{\xi}}}\right| = \left|\frac{d\mathbf{x}}{d\boldsymbol{\xi}}\right| \left|\frac{d\boldsymbol{\xi}}{d\tilde{\boldsymbol{\xi}}}\right| = \det\mathbf{J}_1 \cdot \det\mathbf{J}_2. \tag{9.2}$$

Εν ολίγοις, η ρουτίνα συνάρτησης μορφής αποτελείται από τα εξής μέρη: Αρχικά, σε κάθε στοιχείο καλούνται τα σημεία Gauss στο γονικό στοιχείο και απεικονίζονται στο παραμετρικό χωρίο. Για παράδειγμα, για το παραμετρικό χωρίο $\widehat{\Omega}^e = [\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$, οι παραμετρικές

[2] Εκτενής αναφορά στις ρουτίνες που εμπλέκονται στη μοντελοποίηση και επίλυση προβλημάτων συνοριακών τιμών στο υπολογιστικό πακέτο "T-splines for matlab" γίνεται στο Κεφάλαιο 10 και στο ΠΑΡΑΡΤΗΜΑ Α.

συντεταγμένες $(\xi,\eta) \in \widehat{\Omega}^e$ υπολογίζονται από τις συντεταγμένες $(\tilde{\xi},\tilde{\eta}) \in \widetilde{\Omega}^e$ του γονικού στοιχείου ως εξής

$$\xi = \xi_i + (\tilde{\xi}+1)\frac{(\xi_{i+1}-\xi_i)}{2}$$
$$\eta = \eta_j + (\tilde{\eta}+1)\frac{(\eta_{j+1}-\eta_j)}{2}\ . \tag{9.3}$$

Επομένως, η ορίζουσα της αφινικής απεικόνισης είναι ίση με

$$\det\mathbf{J}_2 = \left|\frac{d\boldsymbol{\xi}}{d\tilde{\boldsymbol{\xi}}}\right| = \frac{(\xi_{i+1}-\xi_i)\cdot(\eta_{j+1}-\eta_j)}{4}. \tag{9.4}$$

Στο δεύτερο μέρος, υπολογίζονται στα παραμετρικά σημεία Gauss οι τιμές των συναρτήσεων βάσης και των παραγώγων τους ως προς τις παραμετρικές συντεταγμένες. Τέλος, στο τρίτο μέρος υπολογίζονται οι φυσικές παράγωγοι, ενώ έχει προηγηθεί ο υπολογισμός του ιακωβιανού πίνακα της γεωμετρικής απεικόνισης. Ο ιακωβιανός πίνακας γράφεται ως

$$\mathbf{J}_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \xi} x_{P_i} & \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \xi} y_{P_i} \\ \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \eta} x_{P_i} & \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \eta} x_{P_i} \end{bmatrix} \tag{9.5}$$

όπου (x_{P_i}, y_{P_i}) οι φυσικές συντεταγμένες του σημείου ελέγχου P_i με $i = 1,\dots,n$.

Επομένως, οι παράγωγοι ως προς τις φυσικές συντεταγμένες, με βάση τη σχέση Χ, υπολογίζονται ως εξής

$$\begin{bmatrix} \frac{\partial R_i}{\partial x} \\ \frac{\partial R_i}{\partial y} \end{bmatrix} = \mathbf{J}_1^{-1} \begin{bmatrix} \frac{\partial R_i}{\partial \xi} \\ \frac{\partial R_i}{\partial \eta} \end{bmatrix}. \tag{9.6}$$

ενώ η στοιχειώδης επιφάνεια υπολογίζεται ως

$$dxdy = |\det J_1| d\xi d\eta = |det\mathbf{J}_1| \frac{(\xi_{i+1}-\xi_i)(\eta_{j+1}-\eta_j)}{4} d\tilde{\xi} d\tilde{\eta}. \tag{9.7}$$

Επομένως, για κάθε στοιχείο με e, όπου $e = 1,\dots,n_{el}$ κατασκευάζεται το στοιχείο (a,b) του τοπικού μητρώου στιβαρότητας, τέτοιο ώστε αν $a,b \in \{1,\dots,n_{en}(e)\}$, τότε

$$k_{ab}^e = \oiint_{\Omega^e} \nabla R_a(\boldsymbol{\varphi}^{-1} \circ \mathbf{x}^{-1}(x,y)) \cdot \nabla R_b(\boldsymbol{\varphi}^{-1} \circ \mathbf{x}^{-1}(x,y)) dxdy =$$
$$= \oiint_{\widehat{\Omega}^e} \nabla R_a(\boldsymbol{\varphi}^{-1}(\xi,\eta)) \cdot \nabla R_b(\boldsymbol{\varphi}^{-1}(\xi,\eta))\, |\det\mathbf{J}_1| d\xi d\eta$$

$$= \int_{-1}^{1}\int_{-1}^{1} \nabla R_a(\tilde{\xi},\tilde{\eta}) \cdot \nabla R_b(\tilde{\xi},\tilde{\eta})\, |\det \mathbf{J}_1| \frac{(\xi_{i+1}-\xi_i)(\eta_{j+1}-\eta_j)}{4} d\tilde{\xi} d\tilde{\eta}$$

$$= \sum_{k=1}^{n_{gauss,x}} \sum_{l=1}^{n_{gauss,y}} w_k w_l \nabla R_a(\tilde{\xi}_k,\tilde{\eta}_l) \cdot \nabla R_b(\tilde{\xi}_k,\tilde{\eta}_l)\, |\det \mathbf{J}_1| \frac{(\xi_{i+1}-\xi_i)(\eta_{j+1}-\eta_j)}{4}. \quad (9.8)$$

9.2 Υπολογιστικές διαδικασίες στην Ισογεωμετρική Ανάλυση με τη συνήθη εξαγωγή Bézier

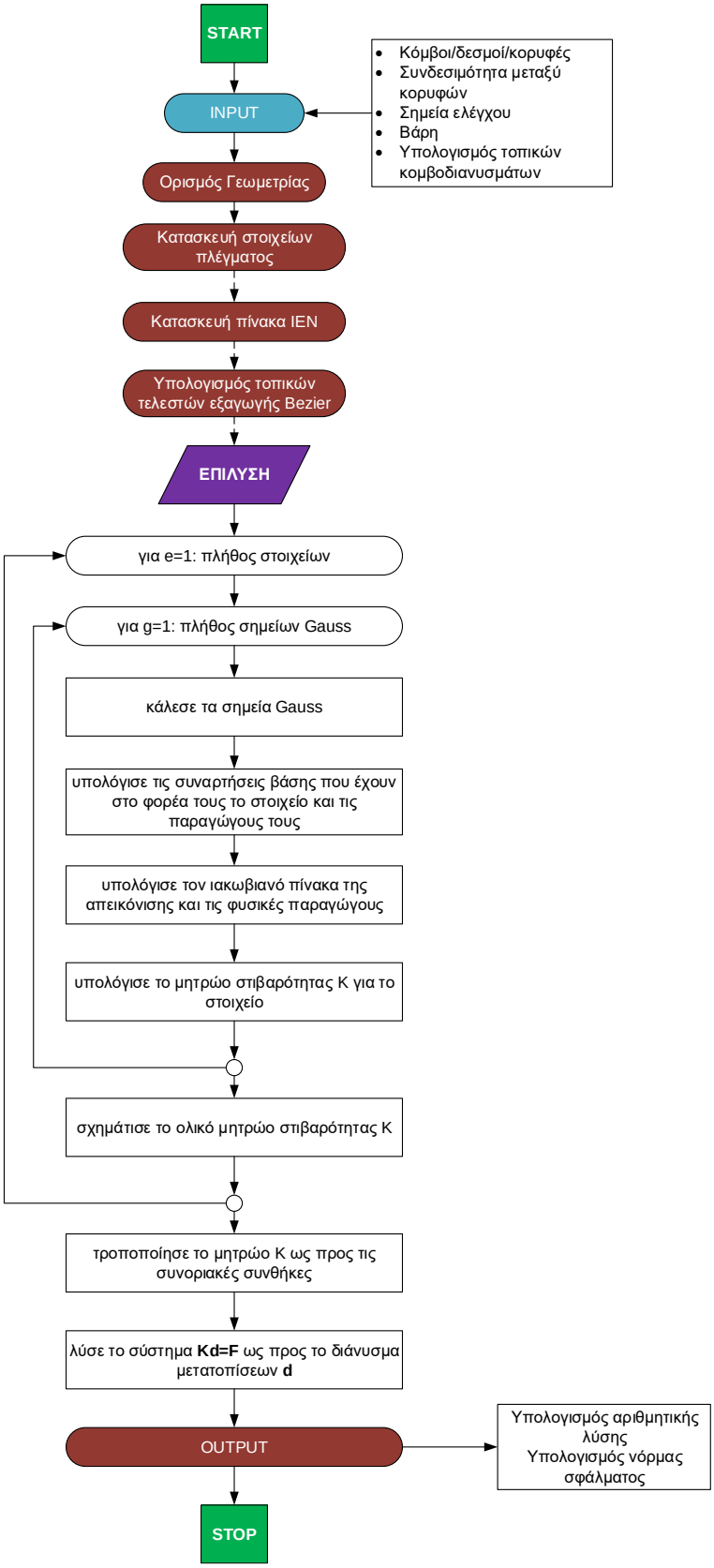
Στην Ισογεωμετρική Ανάλυση με Αποσύνθεση Bézier, η διαδικασία επίλυσης των προβλημάτων συνοριακών τιμών διαφοροποιείται αισθητά. Οι απαιτούμενοι υπολογισμοί σε αυτή την περίπτωση είναι οι παρακάτω:

1. Σχηματίζεται το στοιχειώδες πλέγμα T-spline (elemental T-mesh) και τα επιμέρους στοιχεία (elements) που το απαρτίζουν.
2. Για κάθε στοιχείο του T-spline, σχηματίζεται ο τοπικός τελεστής εξαγωγής Bézier. Για την κατασκευή του, απαιτούνται τα τοπικά κομβοδιανύσματα των συναρτήσεων βάσης με το στοιχείο στο φορέα τους, επομένως και ο πίνακας IEN.
3. Συναρμολογείται το μητρώο στιβαρότητας. Κατά τη συναρμολόγηση του μητρώου, ακολουθούνται τα παρακάτω βήματα για κάθε στοιχείο του T-spline:
   - Καλούνται τα σημεία Gauss του γονικού στοιχείου.
   - Σε κάθε σημείο Gauss υπολογίζονται οι συναρτήσεις βάσης και οι μερικές παράγωγοι τους ως γραμμικός συνδυασμός των πολυωνύμων Bernstein και του τοπικού τελεστή Bézier.
   - Υπολογίζονται ο ιακωβιανός πίνακας και η ορίζουσα της απεικόνισης του γονικού στοιχείου στο φυσικό χωρίο καθώς και οι μερικές παράγωγοι ως προς τις φυσικές συντεταγμένες στα σημεία Gauss.
4. Στο σύστημα εξισώσεων του προβλήματος επιβάλλονται οι συνθήκες Dirichlet, μετασχηματίζοντας το μητρώο στιβαρότητας. Τέλος, το σύστημα στην τελική του εκδοχή επιλύεται ως προς τις μεταβλητές ελέγχου της αριθμητικής λύσης.
5. Η αριθμητική λύση του προβλήματος υπολογίζεται ως γραμμικός συνδυασμός των συναρτήσεων βάσης με συντελεστές τις μεταβλητές ελέγχου.

Με την εξαγωγή Bézier, οι συναρτήσεις βάσης ορίζονται ξεχωριστά για κάθε στοιχείο και έχουν πεδίο ορισμού το στοιχείο αυτό και μόνο. Συνεπώς, οι υπολογισμοί πραγματοποιούνται τοπικά, ανά στοιχείο, όπως και στην Ανάλυση Πεπερασμένων Στοιχείων. Πλέον, δεν είναι απαραίτητη η γνώση των παραμετρικών συντεταγμένων και δεν απαιτείται τα σημεία Gauss να προβάλλονται στον παραμετρικό χώρο για τον υπολογισμό των συναρτήσεων βάσης. Κατά την κατασκευή του μητρώου στιβαρότητας, τα στοιχεία Bézier θα έχουν τον ίδιο ρόλο με τα στοιχεία Lagrange. Επιπλέον, θα καλείται ο αλγόριθμος κατασκευής των συναρτήσεων βάσης (shape function routine) για τον υπολογισμό των συναρτήσεων βάσης και των παραγώγων τους, ενώ η χρήση του πίνακας IEN είναι επιτακτική για την κατασκευή του τελεστή Bézier και αντικαθιστά τον πίνακα συνδεσιμότητας των Πεπερασμένων Στοιχείων.

Στην πλειοψηφία των βιβλιογραφικών αναφορών [21], [6], [11] επισημαίνεται ότι η εξαγωγή Bézier δεν αυξάνει τους βαθμούς ελευθερίας του προβλήματος, καθώς η αριθμητική λύση υπολογίζεται με βάση τις αρχικές συναρτήσεις T-spline. Συγκεκριμένα, στο [21] αναφέρεται ότι τα

σημεία ελέγχου Bézier είναι απλώς εικονικά σημεία ελέγχου και δεν έχουν κάποια φυσική σημασία. Αντιθέτως, χρησιμεύουν μόνο στον ορισμό των στοιχείων Bézier στον φυσικό χώρο και σχετίζονται με τα σημεία ελέγχου της αρχικής γεωμετρίας T-spline/NURBS. Ειδικότερα, στο [11] σημειώνεται ότι το φυσικό πλέγμα Bézier καθορίζει το σύνολο των πεπερασμένων στοιχείων, τα οποία θα χρησιμοποιηθούν στον υπολογισμό των μητρώων στιβαρότητας. Επιπλέον, επισημαίνεται ότι ο τοπικός τελεστής εξαγωγής κάθε στοιχείου χρησιμοποιείται για τον περιορισμό των βαθμών ελευθερίας Bézier έτσι ώστε να διατηρείται η αρχική ομαλότητα των συναρτήσεων T-spline.



Σχήμα 9-2 Αλγόριθμος υπολογισμού της αριθμητικής λύσης σύμφωνα με την Ισογεωμετρική Ανάλυση με εξαγωγή Bézier

9.2.1 Ρουτίνα συνάρτησης μορφής

Στην περίπτωση της Αποσύνθεσης Bézier, η ρουτίνα συνάρτησης μορφής αξιοποιεί τον τελεστή εξαγωγής Bézier που έχει υπολογιστεί προηγουμένως για κάθε στοιχείο. Για τον υπολογισμό των συναρτήσεων βάσης T-spline και των παραμετρικών παραγώγων τους καλείται εντός κάθε στοιχείου η συνάρτηση `TSplineBasis`. Η εν λόγω συνάρτηση βασίζεται στις εξισώσεις (6.31) και (6.33). Εντός της συνάρτησης καλούνται τα πολυώνυμα Bernstein και οι παράγωγοι τους και υπολογίζονται οι συντελεστές βάρους Bezier και οι παράγωγοι της συνάρτησης βάρους.

Καθώς ο τελεστής εξαγωγής Bézier απεικονίζει τις καθολικές συναρτήσεις βάσης σε επίπεδο γονικού στοιχείου, δεν ισχύουν πλέον οι εξισώσεις της απεικόνισης από το γονικό στοιχείο στον παραμετρικό χώρο (9.3) για την εφαρμογή του κανόνα ολοκλήρωσης Gauss. Η απεικόνιση από το γονικό στοιχείο στον παραμετρικό χώρο συνεχίζει να υφίσταται, αλλά πραγματοποιείται πλέον μέσω των τελεστών εξαγωγής Bézier.

Έτσι, ο ιακωβιανός πίνακας της απεικόνισης από το γονικό στο φυσικό χωρίο είναι

$$\begin{aligned}\mathbf{J} = \frac{\partial \tilde{\mathbf{x}}}{\partial \tilde{\xi}} &= (\mathbf{P^e})^T \frac{\partial \mathbf{R}^e}{\partial \tilde{\xi}} \\ &= \left((\mathbf{W}^e)^{-1}(\mathbf{C}^e)^{-T}\mathbf{W}^{b,e}\mathbf{P}^{b,e}\right)^T \frac{\partial \mathbf{R}^e}{\partial \tilde{\xi}}\end{aligned} \tag{9.9}$$

Η ορίζουσα της απεικόνισης στη συμβατική Ισογεωμετρική Ανάλυση είναι ίδια με την ορίζουσα της απεικόνισης στην Ισογεωμετρική Ανάλυση με Εξαγωγή Bézier και η διαφορά τους έγκειται στο γεγονός ότι στην πρώτη περίπτωση χρησιμοποιούνται τα αρχικά σημεία ελέγχου $\mathbf{P}$ του στοιχείου, ενώ στη σχέση (9.9) τα τελικά σημεία ελέγχου, $\mathbf{P}^b$, Bézier.

Ισοδύναμα, η εξίσωση (9.9) γράφεται

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \tilde{\xi}} & \frac{\partial y}{\partial \tilde{\xi}} \\ \frac{\partial x}{\partial \tilde{\eta}} & \frac{\partial y}{\partial \tilde{\eta}} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \tilde{\xi}} x_{P_i} & \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \tilde{\xi}} y_{P_i} \\ \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \tilde{\eta}} x_{P_i} & \sum_{i=1}^{n} \frac{\partial R_{i,p}}{\partial \tilde{\eta}} x_{P_i} \end{bmatrix} \tag{9.10}$$

όπου $\left(x_{P_i}, y_{P_i}\right)$ οι φυσικές συντεταγμένες του σημείου ελέγχου P_i με $i = 1, \dots, n$ και $\frac{\partial R_i}{\partial \tilde{\xi}}$ οι παράγωγοι ως προς τις τοπικές συντεταγμένες του γονικού στοιχείου που υπολογίζονται σύμφωνα με την εξίσωση (6.33).

Ο υπολογισμός των φυσικών παραγώγων πραγματοποιείται υπολογίζοντας αρχικά τον αντίστροφο πίνακα, $\mathbf{J}^{-1}$ κι έπειτα εφαρμόζοντας την παρακάτω σχέση

$$\begin{bmatrix} \frac{\partial R_i}{\partial x} \\ \frac{\partial R_i}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial R_i}{\partial \tilde{\xi}} \\ \frac{\partial R_i}{\partial \tilde{\eta}} \end{bmatrix}, \tag{9.11}$$

Σύμφωνα με τις εξισώσεις της Εξαγωγής Bézier, η στοιχειώδης επιφάνεια υπολογίζεται ως

$$dxdy = |\det\mathbf{J}|d\tilde{\xi}d\tilde{\eta}. \tag{9.12}$$

Επομένως, για κάθε στοιχείο με e, όπου $e = 1, \dots, n_{el}$ με $a, b \in \{1, \dots, n_{en}(e)\}$, το στοιχείο (a, b) στο τοπικό μητρώο στιβαρότητας διατυπώνεται ως

$$\begin{aligned} k_{ab}^{e} &= \oiint_{\Omega^e} \nabla R_a\big(\tilde{\mathbf{x}}^{-1}(x,y)\big) \cdot \nabla R_b\big(\tilde{\mathbf{x}}^{-1}(x,y)\big) dxdy = \\ &= \int_{-1}^{1}\int_{-1}^{1} \nabla R_a\big(\tilde{\xi},\tilde{\eta}\big) \cdot \nabla R_b\big(\tilde{\xi},\tilde{\eta}\big)|\det\mathbf{J}|d\tilde{\xi}d\tilde{\eta} \\ &= \sum_{k=1}^{n_{gauss,x}} \sum_{l=1}^{n_{gauss,y}} w_k w_l \nabla R_a\big(\tilde{\xi}_k,\tilde{\eta}_l\big) \cdot \nabla R_b\big(\tilde{\xi}_k,\tilde{\eta}_l\big)|\det\mathbf{J}|\,. \end{aligned} \tag{9.13}$$

9.3 Υπολογιστικές διαδικασίες στην Ισογεωμετρική Ανάλυση με την πλήρη εξαγωγή Bézier

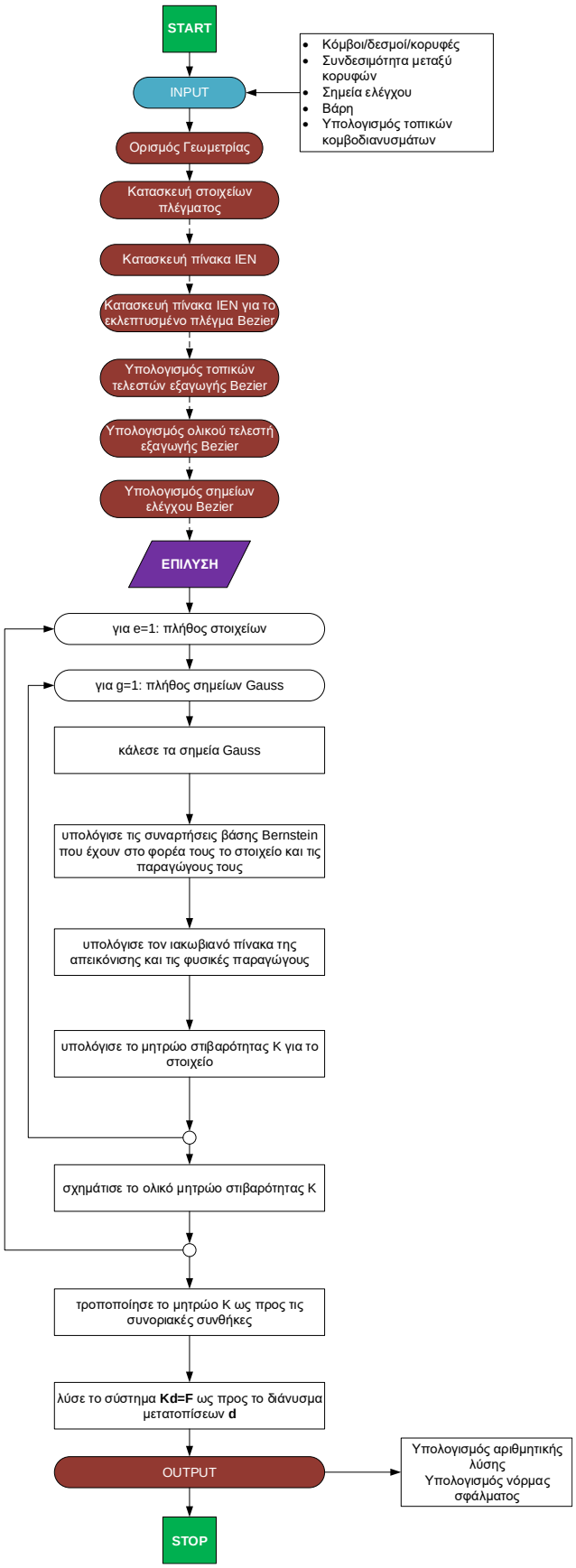
Στην παρούσα εργασία διερευνάται επιπλέον η επίλυση του προβλήματος συνοριακών τιμών με χρήση των τελικών συναρτήσεων βάσης, δηλαδή των (ρητών) πολυωνύμων Bernstein, τα οποία προκύπτουν από την εξαγωγή Bézier. Έτσι, στην επιφάνεια T-spline η οποία πλέον περιγράφεται από πολυώνυμα Bernstein, επιλύεται το πρόβλημα συνοριακών τιμών με πολλαπλάσιους βαθμούς ελευθερίας και μελετάται η επίδραση στην ακρίβεια της αριθμητικής λύσης

Στην προκειμένη περίπτωση, επιλέγουμε να υπολογίσουμε τα τελικά σημεία ελέγχου Bézier και να τα αξιοποιήσουμε στην κατασκευή της αριθμητικής λύσης, η οποία θα αποτελεί γραμμικό συνδυασμό των (ρητών) πολυωνύμων Bernstein. Ο αλγόριθμος επίλυσης διαμορφώνεται όπως παρακάτω:

1. Σχηματίζεται το στοιχειώδες πλέγμα T-spline (elemental T-mesh) και τα επιμέρους στοιχεία (elements) που το απαρτίζουν.
2. Για κάθε στοιχείο του T-spline, σχηματίζεται ο τοπικός τελεστής εξαγωγής Bézier. Για την κατασκευή του, απαιτούνται τα τοπικά κομβοδιανύσματα των συναρτήσεων βάσης με το στοιχείο στο φορέα τους, επομένως και ο πίνακας IEN_1.
3. Κατασκευάζεται το IEN_2 του εκλεπτυσμένου πλέγματος, σύμφωνα με το οποίο δομείται ο ολικός τελεστής εξαγωγής Bézier.
4. Υπολογίζονται τα νέα σημεία ελέγχου Bézier σύμφωνα με τον ολικό τελεστή εξαγωγής Bézier.
5. Συναρμολογείται το μητρώο στιβαρότητας. Κατά τη συναρμολόγηση του μητρώου, ακολουθούνται τα παρακάτω βήματα για κάθε στοιχείο:
   - Καλούνται τα σημεία Gauss του γονικού στοιχείου.
   - Σε κάθε σημείο Gauss υπολογίζονται τα ρητά πολυώνυμα Bernstein και οι μερικές παράγωγοι τους.
   - Υπολογίζονται ο ιακωβιανός πίνακας και η ορίζουσα της απεικόνισης του γονικού στοιχείου στο φυσικό χωρίο καθώς και οι μερικές παράγωγοι ως προς τις φυσικές συντεταγμένες στα σημεία Gauss.

6. Στο σύστημα εξισώσεων του προβλήματος επιβάλλονται οι συνθήκες Dirichlet, μετασχηματίζοντας το μητρώο στιβαρότητας. Τέλος, το σύστημα στην τελική του εκδοχή επιλύεται ως προς τις μεταβλητές ελέγχου της αριθμητικής λύσης.
7. Η αριθμητική λύση του προβλήματος υπολογίζεται ως γραμμικός συνδυασμός των συναρτήσεων βάσης με συντελεστές τις μεταβλητές ελέγχου.

Στην παρούσα εργασία, η προαναφερθείσα μεθοδολογία έχει εφαρμοστεί αποκλειστικά σε πλέγμα NURBS.



Σχήμα 9-3 Αλγόριθμος υπολογισμού της αριθμητικής λύσης σύμφωνα με την Ισογεωμετρική Ανάλυση με εξαγωγή Bézier και χρήση συναρτήσεων Bernstein

9.3.1 Ρουτίνα συνάρτησης μορφής

Εφαρμόζοντας την Αποσύνθεση Bézier κατά τον κλασικό τρόπο, οι νέες συναρτήσεις βάσης του εκλεπτυσμένου πλέγματος είναι τα (ρητά) πολυώνυμα Bernstein, τα οποία χρησιμοποιούνται μαζί με τους τελεστές εξαγωγής για τον υπολογισμό των συναρτήσεων NURBS/T-spline. Δηλαδή, η Αποσύνθεση Bézier προσφέρει έναν εναλλακτικό τρόπο υπολογισμού των αρχικών συναρτήσεων, χωρίς αύξηση των βαθμών ελευθερίας και χωρίς να επηρεάζεται η ομαλότητα της λύσης.

Μια διαφορετική προσέγγιση επιβάλλει τη χρήση απ' ευθείας των συναρτήσεων Bézier του εκλεπτυσμένου πλέγματος. Καθώς, η Αποσύνθεση Bézier είναι επί της ουσίας εκλέπτυνση του πλέγματος NURBS/T-spline με διαδοχική προσθήκη κόμβων στα τοπικά κομβοδιανύσματα, τότε η ιδέα χρήσης των νέων συναρτήσεων στην επίλυση του προβλήματος συνοριακών τιμών έπεται φυσικά.

Έχοντας ήδη υπολογίσει τα μητρώα των τοπικών τελεστών εξαγωγής $\mathbf{C}^e$, απομένει να κατασκευαστεί το μητρώο IEN του εκλεπτυσμένου πλέγματος, ώστε να συναρμολογηθεί σωστά ο τελεστής εξαγωγής Bézier για ολόκληρο το πλέγμα, μέσω του οποίου υπολογίζονται τα νέα σημεία ελέγχου Bézier και οι συντελεστές βάρους Bézier. Επιπλέον, οι συναρτήσεις βάσης είναι τα ρητά ή μη πολυώνυμα Bernstein, τα οποία είναι ήδη γνωστά καθώς υπολογίζονται από τις σχέσεις (6.1) και (6.6), ενώ οι παράγωγοι ως προς τις γονικές συντεταγμένες από τη σχέση (6.7).

Έτσι, ο ιακωβιανός πίνακας της απεικόνισης από το γονικό στο φυσικό χωρίο είναι

$$\mathbf{J} = \frac{\partial \tilde{\mathbf{x}}}{\partial \tilde{\xi}} = \left(\mathbf{P}^{b,e}\right)^T \frac{\partial \mathbf{R}^{b,e}}{\partial \tilde{\xi}} \tag{9.14}$$

όπου $\mathbf{R}^{b^e}$ τα ρητά πολυώνυμα Bernstein που ορίζονται για το στοιχείο Bézier Ω^e και $\mathbf{P}^{b,e}$ τα αντίστοιχα σημεία ελέγχου.

Ισοδύναμα, η εξίσωση (9.14) γράφεται

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \tilde{\xi}} & \frac{\partial y}{\partial \tilde{\xi}} \\ \frac{\partial x}{\partial \tilde{\eta}} & \frac{\partial y}{\partial \tilde{\eta}} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial R_{i,p}^b}{\partial \tilde{\xi}} x_{P_i}^b & \sum_{i=1}^{n} \frac{\partial R_{i,p}^b}{\partial \tilde{\xi}} y_{P_i}^b \\ \sum_{i=1}^{n} \frac{\partial R_{i,p}^b}{\partial \tilde{\eta}} x_{P_i} & \sum_{i=1}^{n} \frac{\partial R_{i,p}^b}{\partial \tilde{\eta}} x_{P_i} \end{bmatrix} \tag{9.15}$$

όπου $\left(x_{P_i}^b, y_{P_i}^b\right)$ οι φυσικές συντεταγμένες του σημείου ελέγχου Bézier $\mathbf{P}_i^b$ με $i = 1, \dots, n$ και $\frac{\partial R_i^b}{\partial \tilde{\xi}}$ οι παράγωγοι των συναρτήσεων Bézier ως προς τις τοπικές συντεταγμένες του γονικού στοιχείου.

Για τον υπολογισμό των φυσικών παραγώγων, ακολουθείται μια ανάλογη διαδικασία με εκείνες που προηγήθηκαν στις προαναφερθείσες μεθοδολογίες. Συγκεκριμένα, υπολογίζεται αρχικά ο αντίστροφος πίνακας, $\mathbf{J}^{-1}$ κι έπειτα εφαρμόζεται η παρακάτω σχέση

$$\begin{bmatrix} \dfrac{\partial R_i^b}{\partial x} \\ \dfrac{\partial R_i^b}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \dfrac{\partial R_i^b}{\partial \tilde{\xi}} \\ \dfrac{\partial R_i^b}{\partial \tilde{\eta}} \end{bmatrix}, \tag{9.16}$$

Σύμφωνα με τα παραπάνω, η στοιχειώδης επιφάνεια υπολογίζεται ως

$$dxdy = |\det\mathbf{J}| d\tilde{\xi} d\tilde{\eta}. \tag{9.17}$$

Επομένως, για κάθε στοιχείο με e, όπου $e = 1, \dots, n_{el}$ με $a, b \in \{1, \dots, n_{en}(e)\}$, το στοιχείο (a, b) στο τοπικό μητρώο στιβαρότητας διατυπώνεται ως

$$\begin{aligned} k_{ab}^e &= \oiint_{\Omega^e} \nabla R_a^b\big(\tilde{\mathbf{x}}^{-1}(x,y)\big) \cdot \nabla R_b^b\big(\tilde{\mathbf{x}}^{-1}(x,y)\big) dxdy = \\ &= \int_{-1}^{1}\int_{-1}^{1} \nabla R_a^b\big(\tilde{\xi},\tilde{\eta}\big) \cdot \nabla R_b^b\big(\tilde{\xi},\tilde{\eta}\big) |\det\mathbf{J}| d\tilde{\xi} d\tilde{\eta} \\ &= \sum_{k=1}^{n_{gauss,x}} \sum_{l=1}^{n_{gauss,y}} w_k w_l \nabla R_a^b\big(\tilde{\xi}_k,\tilde{\eta}_l\big) \cdot \nabla R_b^b\big(\tilde{\xi}_k,\tilde{\eta}_l\big) |\det\mathbf{J}|\,. \end{aligned} \tag{9.18}$$

10 ΥΠΟΛΟΓΙΣΤΙΚΌ ΠΑΚΈΤΟ "T-SPLINES FOR MATLAB"

Η γεωμετρική σχεδίαση των πλεγμάτων T-spline για τα προβλήματα που διερευνώνται στην παρούσα εργασία γίνεται με τη χρήση του υπολογιστικού πακέτου "T-splines for matlab" του Pascal Laube[3]. Το εν λόγω πακέτο υλοποιεί τις συναρτήσεις, οι οποίες περιγράφονται στο άρθρο του Sederberg κ.ά. [5], όπου εισήχθη για πρώτη φορά η έννοια του T-spline, στο προγραμματιστικό περιβάλλον MATLAB. Με βάση τα δεδομένα που εισάγει ο χρήστης, μπορεί να υπολογιστεί και να σχεδιαστεί η προ-εικόνα του T-spline και η εικόνα του στο φυσικό χώρο. Επιπλέον, υποστηρίζεται η προσθήκη κόμβου στην κατεύθυνση ξ. Οι υπόλοιπες δυνατότητες του υπολογιστικού πακέτου ξεφεύγουν από τους σκοπούς της παρούσας εργασίας και ο ενδιαφερόμενος αναγνώστης παροτρύνεται να επισκεφτεί τη σελίδα του δημιουργού. Ο εντοπισμός του υπολογιστικού πακέτου οφείλεται στον υποψήφιο διδάκτορα, κο Ι. Δημητρίου. [22]

Καθώς ο σκοπός δημιουργίας του πακέτου δεν αφορούσε την επίλυση προβλημάτων διαφορικών εξισώσεων σε πλέγματα T-spline με την ισογεωμετρική μέθοδο, κρίθηκε απαραίτητη η επέμβαση στον αρχικό κώδικα με σκοπό τον εμπλουτισμό του με επιπρόσθετες συναρτήσεις που εξυπηρετούν την εφαρμογή της Ισογεωμετρικής Ανάλυσης. Οι τροποποιήσεις αυτές αφορούν τόσο τη διεύρυνση των τοπολογιών που μπορεί να διαχειριστεί το πακέτο, με την προσθήκη συντελεστών βάρους στο πλέγμα, επιτρέποντας τη μελέτη καμπύλων χωρίων, όσο και τον υπολογισμό μεγεθών, όπως είναι το μητρώο στιβαρότητας, τα οποία απαιτούνται για την πραγματοποίηση της ανάλυσης του προβλήματος στα κατασκευασμένα T-spline. Αξίζει να σημειωθεί ότι η δομή του κώδικα επιτρέπει τον ορισμό και τη μελέτη διδιάστατων T-spline αυθαίρετου βαθμού. Η παρούσα εργασία εστιάζει αποκλειστικά σε διδιάστατα T-spline τρίτου βαθμού και δεν έχει διερευνηθεί η αξιοπιστία του κώδικα σε επιφάνειες T-spline διαφορετικού βαθμού.

10.1 Οργάνωση αρχικού υπολογιστικού πακέτου

Προτού μελετηθούν οι επιμέρους τροποποιήσεις, θα παρουσιαστεί η δομή του αρχικού κώδικα και της οργάνωσης, η οποία διέπει τη λειτουργία του. Ο σκελετός του υπολογιστικού πακέτου είναι η *κλάση* `tSpline`. Μέσα σε αυτή ορίζονται ως *ιδιότητες* (properties) της κλάσης, τα επιμέρους χαρακτηριστικά που κατασκευάζουν την επιφάνεια T-spline. Στην αρχική εκδοχή, οι ιδιότητες αυτές αφορούν μόνο τον πολυωνυμικό βαθμό του T-spline, ο οποίος θεωρείται κοινός σε κάθε διάσταση και τις συντεταγμένες των κορυφών του πλέγματος του. Έτσι, η κλάση έχει για τα εν λόγω χαρακτηριστικά του T-spline τις ιδιότητες `deg` και `kVertices`, αντίστοιχα. Ο υπολογισμός των ζητούμενων μεγεθών, για την κατασκευή του παραμετρικού χώρου και την κατασκευή του φυσικού χωρίου κλπ. πραγματοποιείται με τη χρήση συναρτήσεων που βρίσκονται αποθηκευμένες εντός της κλάσης στο πεδίο των *μεθόδων* (methods). Έτσι, λοιπόν, έχουμε μεταξύ άλλων τις ακόλουθες μεθόδους:

Μέθοδος	**Περιγραφή**
`printPreImageColor(obj)/` **`printPreImagePaper(obj)`**	Αφορούν τη σχεδίαση της προ-εικόνας του T-spline.

[3] https://github.com/PascalLaube/tspline

P = evaluate(obj, s, t)	Υπολογίζει τις τιμές των φυσικών συντεταγμένων (x, y) του T-spline για τα σημεία (s, t) του παραμετρικού χώρου.
insertTKnot(obj,s,leftVert,rightVert)	Εισάγει έναν κόμβο ανάμεσα στις κορυφές `leftVert` και `rightVert` στην κατεύθυνση s.

Για την πραγματοποίηση των παραπάνω υπολογισμών, στο πακέτο βρίσκουμε, επίσης τις κλάσεις `kVertex, contrPoint` και `hEdge`. Στην κλάση `kVertex` ορίζονται ως χαρακτηριστικά για την εκάστοτε κορυφή (vertex) τα ακόλουθα δεδομένα:

- Οι συντεταγμένες της s και t στο παραμετρικό χωρίο
- το τοπικό κομβοδιάνυσμα ανά κατεύθυνση που αντιστοιχεί στην κορυφή
- το σημείο ελέγχου της κορυφής και
- οι πλευρές του πλέγματος που συνδέονται άμεσα με την κορυφή. Δηλαδή, η τοπολογική σχέση (συνδεσιμότητα) της κορυφής με εκείνες που βρίσκονται πάνω, κάτω, αριστερά και δεξιά από εκείνη.

Αντίστοιχα, η κλάση `contrPoint` περιέχει ως ιδιότητες της τρεις φυσικές συντεταγμένες x, y, z του σημείου ελέγχου. Επίσης, στην κλάση `hEdge` βρίσκουμε τις ιδιότητες `startVert`, `endVert`, που αναλογούν στην κορυφή από την οποία ξεκινάει και στην οποία καταλήγει, αντίστοιχα, η εκάστοτε ακμή του πλέγματος T-spline.

Για την καλύτερη κατανόηση της δομής του κώδικα, αναλύεται παρακάτω εκτενώς το απλούστερο παράδειγμα T-spline τρίτου βαθμού, δηλαδή η επιφάνεια Bézier τρίτου βαθμού. Η υλοποίηση βασίζεται, σαφώς, στην κλάση `tspline`.

Η εν λόγω επιφάνεια Bézier αποτελεί ένα T-spline –δηλαδή, στο πλαίσιο του αλγορίθμου, είναι ένα *αντικείμενο* (object) της κλάσης `tspline` (class instance). Για την κατασκευή του δημιουργείται ένα αρχείο script, στο οποίο γίνεται η ανάθεση του αντικειμένου στη μεταβλητή `t`. Η κατασκευή του αντικειμένου `t` πραγματοποιείται με τον ίδιο τρόπο που καλείται μια συνάρτηση της MATLAB, δηλαδή, με την κλήση της εντολής `t = tSpline([],3)`, στις πρώτες γραμμές του script. Εδώ, ο αριθμός 3 αντιστοιχεί στο βαθμό της επιφάνειας Bézier, ενώ το κενό διάνυσμα `[]` αντιστοιχεί στις κορυφές (`kvertices`), οι οποίες δεν έχουν ακόμη προσδιοριστεί.

Το παραμετρικό χωρίο μιας επιφάνειας Bézier τρίτου βαθμού αποτελείται από δύο κομβοδιανύσματα σε κάθε κατεύθυνση, τα οποία δεν περιέχουν εσωτερικά σημεία (breakpoints). Με άλλα λόγια, αν θεωρήσουμε ότι το παραμετρικό χωρίο είναι το χωρίο $[0,1] \times [0,1]$, το κομβοδιάνυσμα στην κατεύθυνση ξ θα είναι το $[0,0,0,1,1,1,1]$ κι αντίστοιχα το ίδιο θα είναι και για την κατεύθυνση η. Έτσι, στο T-mesh θα έχουμε τέσσερα σημεία ελέγχου ανά κατεύθυνση, δηλαδή συνολικά δεκαέξι σημεία ελέγχου και αντίστοιχα στο index T-mesh, θα έχουμε δεκαέξι κόμβους/κορυφές. Έτσι, χρειαζόμαστε τα άκρα 0,1 (που είναι και οι μοναδικές κομβικές τιμές) να εμφανιστούν στο index T-mesh με πολλαπλότητα 2. Οι κορυφές του T-mesh έχουν τις ακόλουθες συντεταγμένες:
$(0,0), (0,0), (0,0), (0,0), (1,0), (1,0), (1,0), (1,0)(0,1), (0,1), (0,1), (0,1), (1,1), (1,1), (1,1), (1,1)$.

Η σειρά των παραπάνω σημείων υποδηλώνει και τον τρόπο που απαριθμούμε τις κορυφές, δηλαδή από την κάτω αριστερή κορυφή προς την πάνω δεξιά κορυφή.

Η εκχώρηση των παραπάνω κορυφών στο αντικείμενο `t` της κλάσης `tSpline` γίνεται με την κατασκευή ενός διανύσματος `t.kVertices` μήκους 16 θέσεων, όπου σε κάθε θέση του τοποθετείται ένα αντικείμενο της κλάσης `kVertex` με ορίσματα τις συντεταγμένες της εκάστοτε κορυφής. Πιο απλά,

```
vertices = [ kVertex(0,0)     %vertex 1

             kVertex(0,0)     %vertex 2

             kVertex(0,0)     %vertex 3

             kVertex(0,0)     %vertex 4

             kVertex(1,0)     %vertex 5

             kVertex(1,0)     %vertex 6

             kVertex(1,0)     %vertex 7

             kVertex(1,0)     %vertex 8

             kVertex(0,1)     %vertex 9

             kVertex(0,1)     %vertex 10

             kVertex(0,1)     %vertex 11

             kVertex(0,1)     %vertex 12

             kVertex(1,1)     %vertex 13

             kVertex(1,1)     %vertex 14

             kVertex(1,1)     %vertex 15

             kVertex(1,1)];   %vertex 16


t.kVertices=vertices;
```

Με τις παραπάνω δηλώσεις, δημιουργήσαμε δεκαέξι αντικείμενα `kVertex` στα οποία δώσαμε ως `s` και `t` τις κομβικές συντεταγμένες και τα αποθηκεύσαμε στο διάνυσμα `vertices`, ενώ στη συνέχεια το διάνυσμα `vertices` αποθηκεύτηκε στην ιδιότητα `t.kVertices` του αντικειμένου `t`.

Ενδεικτικά, αν τυπώσουμε την πρώτη κορυφή, γράφοντας `t.kVertices(1)` θα πάρουμε

```
>> t.kVertices(1)


ans =


  kVertex with properties:


          s: 0

          t: 0

    knotVec: []

     cPoint: []

        top: []

      right: []

     bottom: []

       left: []
```

Τα κενά διανύσματα στις ιδιότητες του αντικειμένου `k.Vertex` που εκτυπώθηκε υποδεικνύουν ότι απομένουν να οριστούν για το αντικείμενο τα τοπικά κομβοδιανύσματα `knotVec` ανά κατεύθυνση, το σημείο ελέγχου `cPoint`, καθώς και οι ακμές που συνδέονται με αντικείμενο `top`, `right`, `bottom`, `left`.

10.1.1 Σχεδιασμός τοπολογίας του T-spline

Ο ορισμός των ακμών, με τις οποίες ενώνεται κάθε κορυφή γίνεται με τη χρήση των συναρτήσεων `cnctTop`, `cnctRight`, `cnctBottom`, `cnctLeft`. Έτσι, αν θέλουμε να δηλώσουμε ότι η κορυφή 2 βρίσκεται *δεξιά* της κορυφής 1, γράφουμε

```
vertices(1).cnctRight(vertices(2));
```

Αντίστοιχα, η δήλωση ότι η κορυφή 5 βρίσκεται ακριβώς πάνω από την κορυφή 1 εκφράζεται ως

```
vertices(1).cnctTop(vertices(5));
```

Αξίζει να τονιστεί ότι η δήλωση «η κορυφή 2 βρίσκεται δεξιά της κορυφής 1» δεν συνεπάγεται αυτόματα ότι «η κορυφή 1 βρίσκεται αριστερά της κορυφής 2». Ως εκ τούτου, θα πρέπει να γράψουμε

```
vertices(2).cnctLeft(vertices(1));
```

παρ' ότι φαίνεται πλεονασμός.

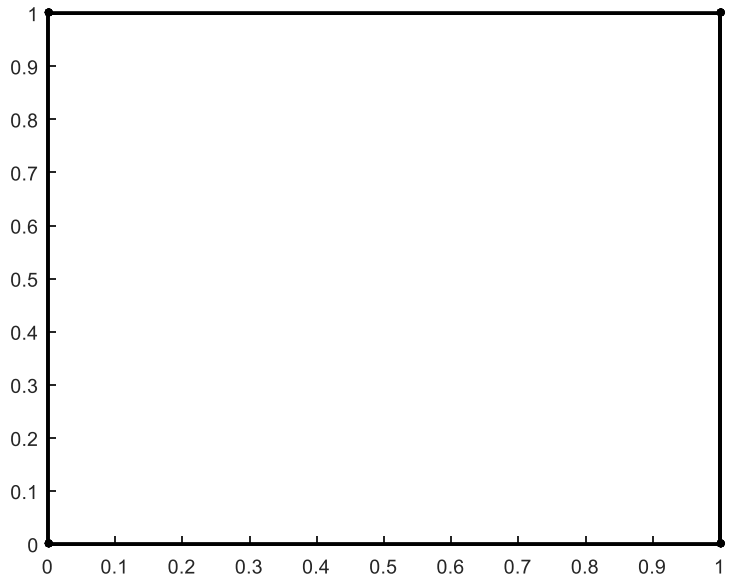
Ακολουθώντας την παραπάνω διαδικασία για όλες τις κορυφές, ορίζουμε την τοπολογική σχέση που υπάρχει μεταξύ τους και η οποία στο παράδειγμα Bézier που μελετάμε, περιγράφεται στον κώδικα με τις παρακάτω εκφράσεις:

```
%Edges

%-----------------------------------

vertices(1).cnctRight(vertices(2));

vertices(1).cnctTop(vertices(5));

%---

vertices(2).cnctTop(vertices(6));

vertices(2).cnctRight(vertices(3));

vertices(2).cnctLeft(vertices(1));

%---

vertices(3).cnctTop(vertices(7));

vertices(3).cnctRight(vertices(4));

vertices(3).cnctLeft(vertices(2));

%---

vertices(4).cnctTop(vertices(8));

vertices(4).cnctLeft(vertices(3));

%---

vertices(5).cnctTop(vertices(9));

vertices(5).cnctRight(vertices(6));

vertices(5).cnctBottom(vertices(1));

%---

vertices(6).cnctTop(vertices(10));
```

```
vertices(6).cnctRight(vertices(7));
vertices(6).cnctBottom(vertices(2));
%---
vertices(7).cnctTop(vertices(11));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctBottom(vertices(3));
vertices(7).cnctLeft(vertices(6));
%---
vertices(8).cnctTop(vertices(12));
vertices(8).cnctBottom(vertices(4));
vertices(8).cnctLeft(vertices(7));
%---
vertices(9).cnctTop(vertices(13));
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(5));
%---
vertices(10).cnctTop(vertices(14));
vertices(10).cnctBottom(vertices(6));
vertices(10).cnctLeft(vertices(9));
vertices(10).cnctRight(vertices(11));
%---
vertices(11).cnctTop(vertices(15));
vertices(11).cnctRight(vertices(12));
vertices(11).cnctBottom(vertices(7));
vertices(11).cnctLeft(vertices(10));   %OK
%---
vertices(12).cnctTop(vertices(16));
```

```
vertices(12).cnctBottom(vertices(8));

vertices(12).cnctLeft(vertices(11));

%---

vertices(13).cnctRight(vertices(14));

vertices(13).cnctBottom(vertices(9));

%---

vertices(14).cnctRight(vertices(15));

vertices(14).cnctBottom(vertices(10));

vertices(14).cnctLeft(vertices(13));

%---

vertices(15).cnctRight(vertices(16));

vertices(15).cnctBottom(vertices(11));

vertices(15).cnctLeft(vertices(14));

%---

vertices(16).cnctLeft(vertices(15));

vertices(16).cnctBottom(vertices(12));

%---


t.kVertices = vertices;
```

Πλέον, έχοντας τις κορυφές και τις ακμές του πλέγματος, μπορούμε να τυπώσουμε την προεικόνα του T-spline στο παραμετρικό χωρίο, χρησιμοποιώντας την εντολή `t.printPreImagePaper();`.



Σχήμα 10-1 Η προ-εικόνα στο παραμετρικό χωρίο της επιφάνειας Bézier

Λόγω των διπλών κόμβων ανά κατεύθυνση, οι παράλληλες πλευρές του πλέγματος ταυτίζονται, δημιουργώντας την εντύπωση ότι το παραμετρικό χωρίο είναι ένα μόνο τετραγωνικό χωρίο με μονούς κόμβους.

10.1.2 Ορισμός σημείων ελέγχου

Ο ορισμός των σημείων ελέγχου ανά κορυφή γίνεται αποδίδοντας στην ιδιότητα `cPoint` κάθε κορυφής `kVertex` ένα διάνυσμα `[x;y;z]`, όπου `x,y,z` οι φυσικές συντεταγμένες του σημείου ελέγχου.

Στην προκειμένη περίπτωση, γνωρίζουμε, από την ισοπαραμετρική αρχή, ότι τα σημεία ελέγχου για την κατεύθυνση ξ είναι τα $(0,0,1)$, $(\frac{1}{3},0,1)$, $(\frac{2}{3},0,1)$ και $(1,0,1)$, όπως και για την κατεύθυνση η (η συντεταγμένη $z=1$, δηλώνει ότι το πλέγμα ελέγχου βρίσκεται στο επίπεδο $z=1$ και δεν παίζει σημαντικό ρόλο στο συγκεκριμένο πλαίσιο). Επομένως, το πλέγμα των δεκαέξι σημείων ελέγχου συνίσταται από το καρτεσιανό γινόμενο των παραπάνω σημείων.

Για να αποδώσουμε, για παράδειγμα, στην κορυφή 5 το σημείο ελέγχου με συντεταγμένες $\left(0,\frac{1}{3},1\right)$, αρκεί να γράψουμε στο αρχείο:

```
vertices(5).cPoint = [0  ;1/3  ;1];
```

Έτσι, εφαρμόζοντας την παραπάνω διαδικασία σε κάθε δεσμό, παίρνουμε τις εξής δηλώσεις:

```
%Control Points

%-----------------------------------

vertices(1).cPoint = [0  ;0  ;1];
```

```
vertices(2).cPoint = [1/3;0  ;1];

vertices(3).cPoint = [2/3;0  ;1];

vertices(4).cPoint = [1  ;0  ;1];


vertices(5).cPoint = [0  ;1/3  ;1];

vertices(6).cPoint = [1/3;1/3  ;1];

vertices(7).cPoint = [2/3;1/3  ;1];

vertices(8).cPoint = [1  ;1/3  ;1];


vertices(9).cPoint = [0  ;2/3  ;1];

vertices(10).cPoint = [1/3;2/3  ;1];

vertices(11).cPoint = [2/3;2/3  ;1];

vertices(12).cPoint = [1  ;2/3  ;1];


vertices(13).cPoint = [0  ;1  ;1];

vertices(14).cPoint = [1/3;1  ;1];

vertices(15).cPoint = [2/3;1  ;1];

vertices(16).cPoint = [1  ;1  ;1];
```

Τέλος, απομένει ο ορισμός των τοπικών κομβοδιανυσμάτων κάθε δεσμού. Ο παραπάνω υπολογισμός πραγματοποιείται με τη συνάρτηση `t.updateKnotVecs();`. Με την συγκεκριμένη συνάρτηση, κατασκευάζονται για κάθε δεσμό τα δύο τοπικά κομβοδιανύσματα, ένα για κάθε διάσταση. Επομένως, αν επανατυπώσουμε τώρα την κορυφή 1, εκτυπώνονται τα εξής:

```
t.kVertices(1)


ans =


  kVertex with properties:
```

```
          s: 0

          t: 0

    knotVec: [2x5 double]

     cPoint: [3x1 double]

        top: [1x1 hEdge]

      right: [1x1 hEdge]

     bottom: []

       left: []
```

Δηλαδή, πλέον στο αντικείμενο `t.kVertices(1)` έχουν αποθηκευτεί οι εξής επιπλέον ιδιότητες

- στο χαρακτηριστικό `knotVec` έχει αποθηκευτεί ένας πίνακας διαστάσεων $2 \times$
- 5, όπου η πρώτη γραμμή του αντιστοιχεί στο τοπικό κομβοδιάνυσμα για την κατεύθυνση ξ και η δεύτερη γραμμή στο τοπικό κομβοδιάνυσμα για την κατεύθυνση η
- στο χαρακτηριστικό `cPoint` βρίσκεται ένα διάνυσμα που περιέχει τις τρεις συντεταγμένες, x,y,z του σημείου ελέγχου
- στο χαρακτηριστικό `top` αποθηκεύτηκε η ακμή (ως αντικείμενο `hEdge`) που ενώνει την κορυφή 1 με την κορυφή 5
- στο χαρακτηριστικό `right` αποθηκεύτηκε η ακμή (πάλι, ως αντικείμενο `hEdge`) που ενώνει την κορυφή 1 με την κορυφή 2.

Επίσης, τα χαρακτηριστικά `bottom`, `left` παρέμειναν κενά, καθώς η κορυφή 1 δεν έχει κάποια κορυφή αριστερά ή κάτω, με την οποία να συνδέεται.

Αν θέλουμε να εκτυπώσουμε τα τοπικά κομβοδιανύσματα `knotVec` της κορυφής, γράφουμε `t.kVertices(1).knotVec` και παίρνουμε

```
t.kVertices(1).knotVec


ans =


     0     0     0     0     0

     0     0     0     0     0
```

Ενώ, αντίστοιχα, για την εκτύπωση της πλευράς `top`, έχουμε

```
t.kVertices(1).top


ans =


  hEdge with properties:


    startVert: [1x1 kVertex]

      endVert: [1x1 kVertex]
```

το οποίο μάς δείχνει ότι η πλευρά `top` ενώνει την κορυφή `startVert` με την κορυφή `endVert`. Αν θέλουμε να μάθουμε ποιες είναι οι συγκεκριμένες κορυφές, γράφουμε `t.kVertices(1).top.startVert` και `t.kVertices(1).top.endVert` αντίστοιχα και παίρνουμε

```
>>t.kVertices(1).top.startVert


ans =


  kVertex with properties:


          s: 0

          t: 0

    knotVec: [2x5 double]

     cPoint: [3x1 double]

        top: [1x1 hEdge]

      right: [1x1 hEdge]

     bottom: []

       left: []
```

και

```
>> t.kVertices(1).top.endVert


ans =


  kVertex with properties:


          s: 0

          t: 0

    knotVec: [2x5 double]

     cPoint: [3x1 double]

        top: [1x1 hEdge]

      right: [1x1 hEdge]

     bottom: [1x1 hEdge]

       left: []
```

Δηλαδή, τα χαρακτηριστικά των κορυφών 1 και 5, αντίστοιχα.

10.1.3 Υπολογισμός φυσικού χωρίου: Συνάρτηση `evaluate(obj,s,t)`

Έχοντας ορίσει τα σημεία ελέγχου, τα τοπικά κομβοδιανύσματα και τη συνδεσιμότητα για κάθε κορυφή, έχουμε κατασκευάσει τον κορμό της επιφάνειας Bézier. Έχουμε, δηλαδή, όλη την πληροφορία που χρειαζόμαστε για να εκτυπώσουμε το φυσικό χωρίο Bézier, χρησιμοποιώντας τη συνάρτηση `t.evaluate`.

Γενικότερα, με τη συνάρτηση `evaluate(obj,s,t)` υπολογίζουμε τις τιμές της επιφάνειας T-spline στα σημεία `(s,t)` του παραμετρικού χωρίου. Η σχέση που συνδέει το φυσικό χωρίο T με το παραμετρικό είναι η εξής:

$$T(\xi,\eta) = \frac{\sum_{A=1}^{N} \mathbf{P}_A \, N_A(\xi,\eta)}{\sum_{B=1}^{N} N_B(\xi,\eta)} \tag{10.1}$$

όπου P_A τα σημεία ελέγχου του πλέγματος, N_A οι πολυωνυμικές συναρτήσεις ανάμειξης και N ο συνολικός αριθμός των δεσμών ή αλλιώς, οι βαθμοί ελευθερίας του T-spline.

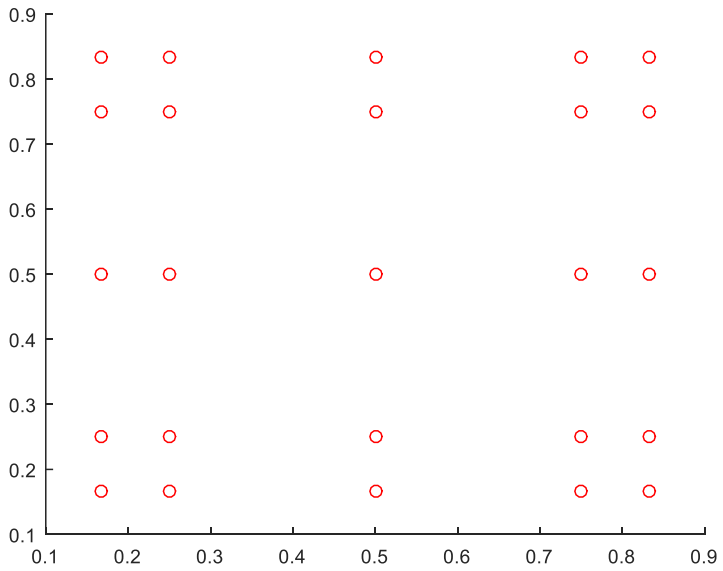
Για την χρήση της συνάρτησης `evaluate`, διαμερίζουμε το παραμετρικό χωρίο ανά κατεύθυνση σε καθορισμένο αριθμό σημείων, πάνω στα οποία καλείται η συνάρτηση.

Έτσι, για παράδειγμα, στην επιφάνεια Bézier, αν η διαμέριση πραγματοποιηθεί σε 5 ομοιόμορφα κατανεμημένα σημεία ανά κατεύθυνση του παραμετρικού χωρίου, θα έχουμε τις εξής εντολές:

```
sampleCount = 5;

ps = linspace(0,1,sampleCount);

pt = linspace(0,1,sampleCount);
```

και η εκτύπωση του γραφήματος του φυσικού χωρίου πραγματοποιείται με τη χρήση της εντολής `plot3`, αφού προηγουμένως έχουν υπολογιστεί οι συντεταγμένες του φυσικού χωρίου με τη συνάρτηση `evaluate`:

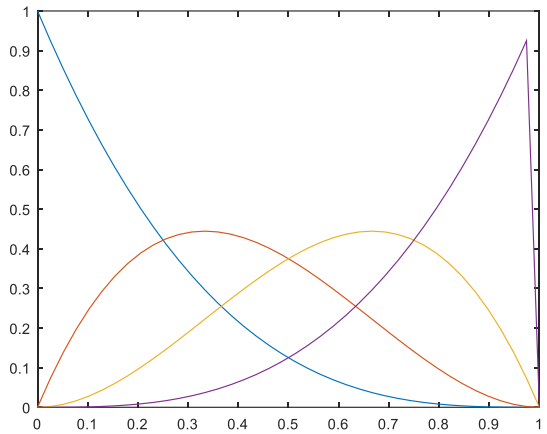
```
hold on

for i=1:sampleCount

    for j=1:sampleCount

            P = t.evaluate(ps(i),pt(j));

            plot3(P(1),P(2),P(3),'ro');

    end

end
```



Σχήμα 10-2 Η επιφάνεια Bézier που σχεδιάζεται με τη διαμέριση του παραμετρικού χωρίου σε πέντε ομοιόμορφα κατανεμημένα σημεία ανά κατεύθυνση.

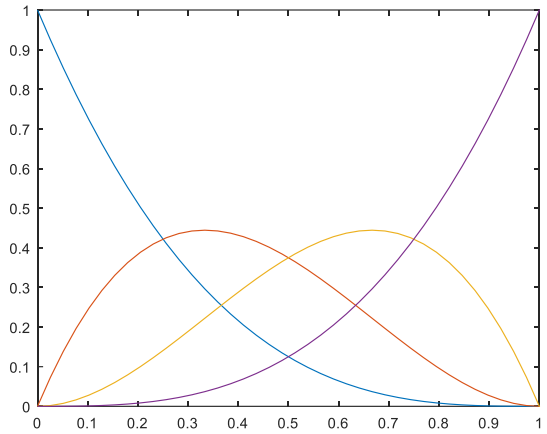
10.1.4 Υπολογισμός συναρτήσεων βάσης ανά κατεύθυνση

Για τον υπολογισμό του φυσικού χωρίου, η συνάρτηση `evaluate` καλεί τη συνάρτηση `[ N ] = basisFunTSpl( x, knots, deg, knotspan`[4]`)`. Η συγκεκριμένη συνάρτηση υπολογίζει στο σημείο `x` τη συνάρτηση βάσης βαθμού `deg` που αντιστοιχεί στο τοπικό κομβοδιάνυσμα `knots`. Για το σκοπό αυτό, καλείται εντός της `basisFunTSpl`, η `basisRecur` η οποία βασίζεται στον αναδρομικό ορισμό των συναρτήσεων βάσης Cox-de Boor [23]. Λόγω του τρόπου κατασκευής της, η `basisRecur` αδυνατεί να υπολογίσει την ακραία τιμή για κάποιες συναρτήσεις βάσης. Στο σχήμα Χ, γίνεται φανερή αυτή η αδυναμία. Πράγματι, αν υπολογίσουμε τις συναρτήσεις βάσης της καμπύλης Bézier για το ολικό κομβοδιάνυσμα [0,0,0,0,1,1,1,1], δηλαδή τα πολυώνυμα Bernstein, θα δούμε ότι η τέταρτη συνάρτηση βάσης στο σημείο $\xi = 1$ μηδενίζεται αντί να πάρει την τιμή 1. Δηλαδή, στο σημείο αυτό το άθροισμα των συναρτήσεων βάσης δεν είναι ίσο με τη μονάδα. Για τη διόρθωση αυτής της αστοχίας, χρησιμοποιείται για τον υπολογισμό των συναρτήσεων βάσης ανά κατεύθυνση η συνάρτηση `basisFunTSpl`, η οποία «επιβάλλει» την τιμή 1 στα άκρα των συναρτήσεων, που κατά παράβαση του κανόνα, μηδενίζονται.



Σχήμα 10-3 Οι συναρτήσεις βάσης (πολυώνυμα Bernstein) που αντιστοιχούν στο κομβοδιάνυσμα $\Xi = [0,0,0,0,1,1,1,1]$ όπως σχεδιάζονται με χρήση της συνάρτησης basisRecur. Στην παραμετρική συντεταγμένη $\xi = 1$, δηλαδή στο δεξιό άκρο, η τέταρτη συνάρτηση $N_{4,3}$ λαμβάνει την τιμή 0, αντί της τιμής 1, με αποτέλεσμα να μην επαληθεύεται η ιδιότητα της διαμέρισης της μονάδας.

[4] Η τιμή `knotspan` αντιστοιχεί στον αύξοντα αριθμό της συνάρτησης βάσης που ορίζεται από το ολικό κομβοδιάνυσμα `knots`. Δεδομένου ότι στα T-splines, εργαζόμαστε μόνο με τοπικά κομβοδιανύσματα, η τιμή `knotspan` παίρνει πάντα την τιμή 1, καθώς σε κάθε τοπικό κομβοδιάνυσμα αντιστοιχεί μόνο μια συνάρτηση βάσης.



Σχήμα 10-4 Με τη συνάρτηση `basisFunTspl` διορθώνεται η αστοχία στα άκρα και εκτυπώνονται οι σωστές συναρτήσεις βάσης.

Σε αυτό το σημείο, αξίζει να τονιστεί ότι ο αρχικός κώδικας του πακέτου δεν εξάγει σε ξεχωριστή ρουτίνα τις συναρτήσεις ανάμειξης του T-Spline. Αντιθέτως, η συνάρτηση `evaluate` χρησιμοποιεί το γινόμενο των συναρτήσεων βάσης ανά κατεύθυνση, αποκλειστικά και μόνο για τον υπολογισμό του φυσικού χωρίου, σύμφωνα με την εξίσωση (10.1).

10.1.5 Εισαγωγή κόμβου

Η εισαγωγή κάθε κόμβου στην κατεύθυνση s πραγματοποιείται με την εφαρμογή της σχέσης(5.3), η οποία συζητήθηκε στο Κεφάλαιο 5.1. Η προσέγγιση της τεχνικής γίνεται με τη χρήση των κομβοδιαστημάτων και ορίζεται στη συνάρτηση `insertTKnot(obj,s,leftVert,rightVert)`, όπου `s` η τιμή του νέου κόμβου, `leftVert` η αριστερή κορυφή και `rightVert` η δεξιά κορυφή του κομβοδιαστήματος εισαγωγής.

Η αντιστοίχιση των μεταβλητών των σχέσεων του Κεφαλαίου 5.1 με εκείνες της συνάρτησης του κώδικα είναι η εξής:

d_0	`d1`
d_1	`d2`
d_2	`d3+d4`
d_L	`d3`
d_R	`d4`
d_3	`d5`
d_4	`d6`

10.2 Εμπλουτισμός του υπολογιστικού πακέτου

Όπως έχει γίνει σαφές, ο κώδικας στην αρχική του μορφή παρουσιάζει κάποιες αδυναμίες, όσον αφορά την αξιοποίηση του για τους σκοπούς της παρούσας εργασίας. Παρ' ότι μάς προσφέρει σημαντικά εργαλεία ώστε να κατασκευάσουμε με εύκολο και λιτό τρόπο μια επιφάνεια T-spline τρίτου βαθμού και με βάση τα επιμέρους χαρακτηριστικά του, αδυνατεί να μας προσφέρει τα αναλυτικά εργαλεία που χρειαζόμαστε για να προχωρήσουμε στην εφαρμογή αριθμητικών μεθόδων επίλυσης προβλημάτων διαφορικών εξισώσεων, μιας και ξεφεύγει των επιδιώξεων του δημιουργού του πακέτου. Επιπλέον,

Ένας από τους στόχους της εργασίας ήταν, λοιπόν, να τροποποιήσει και να επεκτείνει τον υπάρχοντα κώδικα, ώστε να μπορέσει να αξιοποιηθεί στο πλαίσιο της Ισογεωμετρικής Ανάλυσης.

Οι ρουτίνες, που στην πλειονότητα τους προστέθηκαν εντός της κλάσης `tspline` αφορούν τόσο τη βελτίωση της γεωμετρικής απεικόνιση του T-spline όσο και την προσαρμογή του κώδικα στην επίλυση προβλημάτων μηχανικής, όπως της σταθερής εξίσωσης μεταφοράς θερμότητας που θα αναλυθεί εκτενώς στο Κεφάλαιο 11.

10.2.1 Οι συναρτήσεις blendfun, blendfunNorm και blendfunNormW

Η πρώτη προσθήκη αφορά την κατασκευή της ρουτίνας `blendfun` η οποία υπολογίζει αυτόνομα τις πολυωνυμικές συναρτήσεις T-spline, ανεξάρτητα από τη ρουτίνα υπολογισμού του φυσικού χωρίου `evaluate`. Επιπλέον, κατασκευάστηκαν οι ρουτίνες `blendfunNorm` και `blendfunNormW`. Η πρώτη αφορά τον υπολογισμό των ρητών συναρτήσεων T-spline με βάρος 1, και η δεύτερη τον υπολογισμό των ρητών συναρτήσεων με τη χρήση των βαρών που εισήχθησαν από το χρήστη κατά την κατασκευή του T-spline. Σημειώνεται δε σε αυτό το σημείο, ότι ο ορισμός της επιφάνειας T-spline, όπως δόθηκε το 2010 στο [4], λαμβάνει υπόψιν τις ρητές συναρτήσεις και όχι τις πολυωνυμικές. Οι ρητές συναρτήσεις a priori ικανοποιούν την ιδιότητα της διαμέρισης της μονάδας σε κάθε σημείο του χωρίου. Περισσότερα, πάνω σε αυτό το ζήτημα έχουν συζητηθεί στα Κεφάλαια 4 και 7. Επιπλέον, οι παραπάνω κατασκευασμένες συναρτήσεις, υπολογίζουν τις μερικές παραγώγους των συναρτήσεων T-spline ως προς τις παραμετρικές συντεταγμένες, καθώς και τις μονοδιάστατες συναρτήσεις που συνιστούν τις συναρτήσεις T-spline. Για την κατασκευή της ρουτίνας ήταν απαραίτητη η κατασκευή και της συνάρτησης υπολογισμού των μερικών παραγώγων των ανά κατεύθυνση συναρτήσεων B-spline, derRecur, σύμφωνα με την αναδρομική σχέση (2.4).

10.2.2 Η συνάρτηση printcontrolmesh

Επιπλέον, κατασκευάστηκε η συνάρτηση `printcontrolmesh(obj)`, η οποία τυπώνει το πλέγμα των σημείων ελέγχου του T-spline (T-mesh). Στη θεωρία των T-spline, πολλές φορές το πλέγμα ελέγχου (control mesh) χρησιμοποιείται αντί του T-mesh που ορίζεται στο χώρο δεικτών (index T-mesh)., η έννοια T-mesh δηλώνει το πλέγμα σημείων ελέγχου, (πάνω στο οποίο ορίζονται επιπλέον οι κόμβοι ή τα κομβικά διαστήματα) ενώ σε άλλες ως T-mesh ορίζεται σαφώς το index space του T-spline. Σε κάθε περίπτωση, κρίνεται απολύτως σημαντικός ο υπολογισμός του πλέγματος ελέγχου, ως σκελετού του φυσικού χωρίου.

10.2.3 Οι συναρτήσεις patches_calc και elements_calc

Συμπληρωματικά, κρίθηκε αναγκαίος ο αυτόματος υπολογισμός των τμηματικών χωρίων (patches), που σχηματίζονται από τις ακμές του πλέγματος T-spline, δηλαδή οι έδρες του πλέγματος T-spline. Για το λόγο αυτό κατασκευάστηκε η συνάρτηση `patches_calc(obj, nopatches)`, η οποία

εξάγει έναν πίνακα διαστάσεων `nopatches` $\times 8$, όπου σε κάθε γραμμή του καταγράφονται οι συντεταγμένες $x_A, y_A, x_B, y_B, x_C, y_C, x_D, y_D$ του χωρίου με κορυφές $ABCD$ και `nopatches` ο συνολικός αριθμός των τμηματικών χωρίων του πλέγματος.

Στο πλαίσιο των T-spline, τα τμηματικά χωρία του T-mesh (patches ή αλλιώς στη βιβλιογραφία T-mesh elements) δεν ταυτίζονται απαραίτητα με τα στοιχεία (T-spline elements), δηλαδή, με τις τετράπλευρες περιοχές του T-spline, οι οποίες φράσσονται από τις γραμμές μειωμένης συνέχειας και στις οποίες πραγματοποιείται αριθμητική ολοκλήρωση. Ο υπολογισμός των χωρίων αυτών, λοιπόν, δεν είναι επαρκής για την εφαρμογή της ισογεωμετρικής μεθόδου, ως εκ τούτου, κατασκευάστηκε η ρουτίνα `elements_calc(obj, noelmnts)`, η οποία υπολογίζει τις συντεταγμένες των στοιχείων, απαριθμώντας τα από αριστερά και κάτω του παραμετρικού χωρίου με κατεύθυνση προς τα δεξιά και προς τα πάνω.

10.2.4 Η συνάρτηση `printElementalTmesh`

Η κατασκευή των στοιχείων T-spline προϋποθέτει, όμως, και την κατασκευή των γραμμών μειωμένης συνέχειας. Οι γραμμές μειωμένης συνέχειας αποτελούν προεκτάσεις των ακμών που σχηματίζουν διασταύρωση T στο T-spline. Οι γραμμές αυτές τέμνουν κάθετα τις ακμές που συναντούν, δημιουργώντας επιπλέον σημεία. Αυτές οι επιπλέον γραμμές, μαζί με τα νέα σημεία δημιουργούν τα προαναφερθέντα στοιχεία του t-spline. Η υλοποίηση υπολογισμού των σημείων τομών που δημιουργούν οι γραμμές μειωμένης συνέχειας πραγματοποιήθηκε με την προσθήκη της ιδιότητας `fVertices` στην κλάση `tspline` και την κατασκευή της κλάσης `fVertex`. Οι κορυφές `fVertices` διαφέρουν ποιοτικά από τις κορυφές του πλέγματος `kVertices`, καθώς δεν αντιστοιχούν σε πραγματικούς δεσμούς που αναλογούν σε συναρτήσεις T-spline και σημεία ελέγχου. Πρόκειται απλώς για τα επιπρόσθετα σημεία, από τα οποία σχηματίζεται το στοιχειώδες πλέγμα T-spline (elemental T-mesh) και τα οποία ορίζουν τις περιοχές ολοκλήρωσης. Επομένως, στα χαρακτηριστικά της κλάσης `fVertex` προστίθενται μόνο οι παραμετρικές συντεταγμένες της «ψευδο-κορυφής», χωρίς να περιλαμβάνονται οι επιπλέον ιδιότητες της κλάσης `kVertex`, όπως τα τοπικά κομβοδιανύσματα και το σημείο ελέγχου.

Ο υπολογισμός των σημείων `fVertices` πραγματοποιείται με τη χρήση της συνάρτησης `fVertices_last`, με την οποία εξάγονται οι παραμετρικές συντεταγμένες των `fVertices` και οι πλευρές, που δημιουργούνται με την προσθήκη τους, δηλαδή οι γραμμές μειωμένης συνέχειας που δεν υπάρχουν στο αρχικό T-mesh.

Επομένως, προσθέτοντας τις γραμμές μειωμένης συνέχεια και τα σημεία τομής τους με τις ακμές του πλέγματος, έχουμε το στοιχειώδες T-mesh, το οποίο τυπώνουμε, χρησιμοποιώντας την συνάρτηση `printElementalTmesh(obj)`.

10.2.5 Το χαρακτηριστικό `weights` στην κλάση `tspline`

Ο αρχικός κώδικας αδυνατεί να κατασκευάσει και να επεξεργαστεί επιφάνειες T-spline, οι οποίες ορίζονται με τη χρήση συντελεστών βάρους. Με άλλα λόγια, ο αρχικός κώδικας δεν μπορεί να κατασκευάσει επιφάνειες T-spline με καμπυλότητα. Η επίλυση αυτής της δυσκολίας πραγματοποιήθηκε με την προσθήκη του χαρακτηριστικού `weights` στην κλάση `tspline`. Έτσι, όταν ο χρήστης ορίζει τη γεωμετρία του T-spline, θέτοντας τους δεσμούς και τα σημεία ελέγχου, μπορεί να προσθέσει επιπλέον και το βάρος κάθε δεσμού.

Για το σκοπό αυτό, τροποποιήθηκε και η αρχική συνάρτηση `evaluate`, ώστε να λαμβάνει υπόψιν και το βάρος κάθε συνάρτησης T-spline, ενώ η ρουτίνα υπολογισμού των συναρτήσεων T-spline `blendfunNormW` υλοποιεί τον ορισμό των ρητών συναρτήσεων που λαμβάνει υπ' όψιν τα βάρη κάθε συνάρτησης.

10.2.6 Ρουτίνα `BE_Ce` για την εξαγωγή Bézier σε διδιάστατο T-spline

Τέλος, κατασκευάστηκε η ρουτίνα υπολογισμού των διδιάστατων τοπικών τελεστών εξαγωγής Bézier `BE_Ce`. Η ρουτίνα λειτουργεί ανεξάρτητα από την κλάση `tspline` και βασίζεται στον αλγόριθμο που παρουσιάστηκε στο [13] και υπολογίζει τον τοπικό τελεστή εξαγωγής Bézier για μονοδιάστατη συνάρτηση T-spline. Η αλγοριθμική ρουτίνα παρατίθεται στο Παράρτημα Α.5.2.

11 ΥΠΟΛΟΓΙΣΤΙΚΈΣ ΕΦΑΡΜΟΓΈΣ

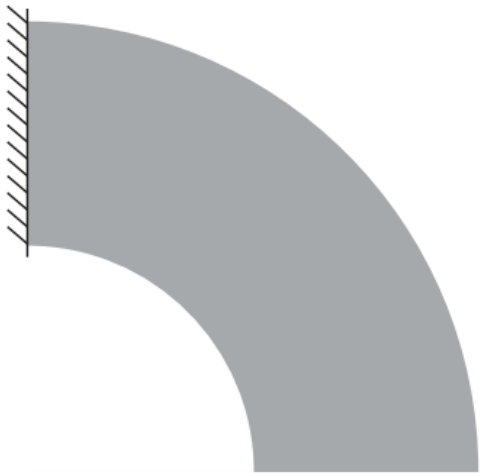
Στο παρόν Κεφάλαιο θα ασχοληθούμε με την επίλυση της εξίσωσης Laplace που περιγράφει τη μόνιμη ροή θερμότητας, αξιοποιώντας το υπολογιστικό πακέτο και τις βελτιώσεις που πραγματοποιήθηκαν σε αυτό. Θα ορίσουμε το πρόβλημα Laplace σε ορθογώνια, αλλά και σε καμπυλόγραμμα χωρία, ώστε να αξιοποιήσουμε στο μέγιστο τις δυνατότητες του βελτιωμένου πακέτου. Επιπλέον, θα πραγματοποιήσουμε εξαγωγή Bézier και θα επιλύσουμε το πρόβλημα της θερμότητας στην επιφάνεια Bézier. Θα μελετήσουμε τα T-spline ως προς την καταλληλόλητα τους για την υλοποίηση μεθόδων ισογεωμετρικής ανάλυσης, ενώ σε περιπτώσεις που είναι εφικτό, θα μετατρέψουμε τις επιφάνειες T-spline σε επιφάνειες NURBS, επιλύοντας εκ νέου την εξίσωση Laplace και συγκρίνοντας τα σφάλματα μεταξύ των δύο προσεγγίσεων. Η γεωμετρίες, στις οποίες θα ορίσουμε τα προβλήματα θα περιγραφούν αποκλειστικά από συναρτήσεις τρίτου βαθμού ($p = 3$) και διάστασης $d_p = 2$. Επίσης, σε όλες τις περιπτώσεις, η νόρμα L_2 του σφάλματος δίνεται από τη σχέση:

$$\|T - T_{exact}\|_2 = \sqrt{\frac{\int_\Omega (T - T_{exact})^2 \, dT}{\int_\Omega T_{exact}^2 \, dT}}, \tag{11.1}$$

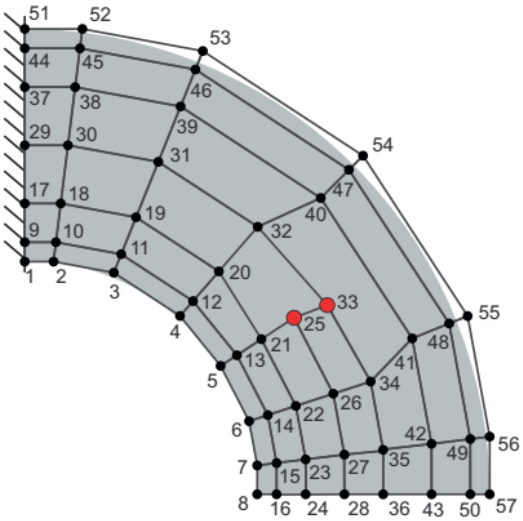
όπου T, η αριθμητική λύση και T_{exact} η θεωρητική λύση.

11.1 Δοκός τεταρτοκυκλίου

Το πρώτο παράδειγμα T-spline που θα εξετάσουμε βασίζεται στη γεωμετρία που περιγράφεται στο [11], από όπου έχει αντληθεί μεγάλο μέρος των ορισμών και της θεωρίας που παρουσιάζονται στο Κεφάλαιο. Πρόκειται για μια δοκό σε σχήμα τεταρτοκυκλίου (Σχήμα 11-1) και μπορεί να περιγραφεί από το κυβικό T-mesh που φαίνεται στο Σχήμα 11-2. Στο συγκεκριμένο παράδειγμα, χρησιμοποιούμε τον όρο T-mesh, εννοώντας τόσο το index T-mesh όσο και το πλέγμα ελέγχου του T-spline.



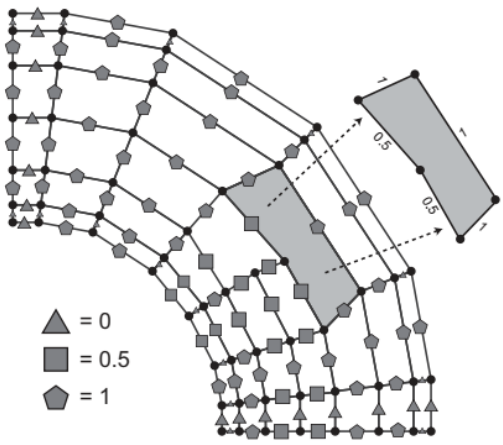
Σχήμα 11-1 Το χωρίο $\Omega \subset \mathbb{R}^2$ ενός διδιάστατου ($d_p = 2$) κυβικού ($p = 3$) T-spline. Τα καμπύλα σύνορα είναι ακριβή τεταρτοκύκλια, στα οποία εφαρμόζονται συνοριακές συνθήκες Dirichlet. [13]



Σχήμα 11-2 Ένα T-mesh το οποίο ορίζει μια κυβική γεωμετρία T-spline. Οι μεγάλοι κύκλοι υποδηλώνουν τις διασταυρώσεις T στο T-mesh. Η αρίθμηση καθορίζει τα σημεία ελέγχου του T-spline. [13]

Οι μαύροι κύκλοι ● αντιστοιχούν στις κορυφές του T-mesh ή ισοδύναμα, στα σημεία ελέγχου. Οι κόκκινοι κύκλοι P_{25} και P_{33} αναπαριστούν τις διασταυρώσεις T. Τονίζουμε ότι η συγκεκριμένη γεωμετρία είναι αρκετά απλή και μπορεί να αναπαρασταθεί πολύ πιο λιτά και συνοπτικά με συναρτήσεις NURBS ή ακόμα και T-spline. Στην περίπτωση των NURBS θα επαρκούσαν 6 σημεία ελέγχου, ενώ στην περίπτωση των κυβικών T-spline, απαιτούνται μόνο 16 σημεία ελέγχου. Τα επιπλέον σημεία ελέγχου στο T-mesh υποδεικνύουν ότι η Ανάλυση Πεπερασμένων Στοιχείων απαιτεί κατά κανόνα μεγαλύτερο πλήθος βαθμών ελευθερίας από εκείνο που απαιτείται για την ακριβή γεωμετρική απεικόνιση του χωρίου. [11]

Στο Σχήμα 11-3 βλέπουμε μια έγκυρη διάταξη κομβικών διαστημάτων στο T-mesh του Σχήμα 11-2. Παρατηρούμε, επίσης, ότι τα ακραία κομβικά διαστήματα περιμετρικά του T-mesh έχουν μήκος 0. Αυτά τα διαστήματα μηδενικού μήκους παίζουν παρόμοιο ρόλο με αυτόν των ανοιχτών κομβοδιανυσμάτων στις συναρτήσεις NURBS και διευκολύνουν την επιβολή συνοριακών συνθηκών. Υπενθυμίζεται ότι ένα κομβοδιάνυσμα NURBS χαρακτηρίζεται ανοιχτό όταν η πολλαπλότητα των ακραίων κόμβων του είναι $p + 1$. Στη μονοδιάστατη περίπτωση, αυτό συνεπάγεται ότι τα δυο ακραία σημεία ελέγχου παρεμβάλλουν την καμπύλη, ενώ στη διδιάστατη περίπτωση ότι κάθε έδρα του παραμετρικού χωρίου είναι επιφάνεια NURBS και κάθε ακμή των επιφανειών αυτών είναι καμπύλη NURBS. Έτσι, η συνοριακή συνάρτηση στο σύνορο Dirichlet μπορεί να παρεμβληθεί με τη χρήση συναρτήσεων NURBS.



Σχήμα 11-3 Μια έγκυρη διάταξη των κομβικών διαστημάτων για το κυβικό T-mesh του Σχήμα 11-2. Τα τρίγωνα αντιστοιχούν σε μηδενικό κομβοδιάστημα, τα τετράγωνα σε κομβοδιάστημα μήκους $\frac{1}{2}$ και τα πεντάγωνα σε κομβοδιάστημα μήκους 1. Μια έγκυρη διάταξη κομβοδιαστημάτων για ένα στοιχείο φαίνεται και στο εξέχον στοιχείο. Επισημαίνεται ότι τα κομβικά διαστήματα των απέναντι ακμών ενός στοιχείου θα πρέπει να αθροίζουν στην ίδια τιμή. [13]

Πριν εμβαθύνουμε στο πρόβλημα Laplace όπως ορίζεται για τη συγκεκριμένη γεωμετρία, θα πρέπει να μελετηθεί κατά πόσο οι συναρτήσεις του συγκεκριμένου T-spline μπορούν να χρησιμοποιηθούν στην Ισογεωμετρική Ανάλυση. Για το σκοπό αυτό, κατασκευάζεται το επεκτεταμένο πλέγμα T-spline. Δηλαδή, προσθέτουμε στο T-mesh τις επεκτάσεις έδρας και ακμής για τις διασταυρώσεις T_{25} και T_{33}. Όπως γίνεται φανερό στο Σχήμα 11-4, οι επεκτάσεις τέμνονται στη διασταύρωση T_{25}, καθιστώντας το πλέγμα μη κατάλληλο-προς-ανάλυση (not analysis-suitable), σύμφωνα με τον ορισμό των Li κ.ά. [15], που δόθηκε στο Κεφάλαιο 4.3. Αυτό φυσικά δεν συνεπάγεται ότι οι συναρτήσεις του T-spline δεν συνιστούν βάση του χώρου T-spline, αλλά ότι η τοπολογία του πλέγματος δεν ανήκει στην κατηγορία εκείνη των T-spline, για τα οποία γνωρίζουμε εκ των προτέρων ότι ικανοποιείται η γραμμική ανεξαρτησία των συναρτήσεων. Επομένως, θα πρέπει να χρησιμοποιήσουμε διαφορετικά εργαλεία για την διερεύνηση της γραμμικής ανεξαρτησίας, όπως είναι ο καθολικός τελεστής Bézier (Κεφάλαιο 7). Ο καθολικός τελεστής Bézier συναρμολογώντας τους επιμέρους τοπικούς τελεστές Bézier σε ένα μητρώο διάστασης $(\text{πλήθος δεσμών}) \times (p+1) \cdot (q+1) \cdot (\text{πλήθος στοιχείων})$. Έτσι, για το εν λόγω χωρίο T-spline, η διάσταση του θα είναι 57×384. Υπολογίζοντας την τάξη του καθολικού τελεστή Bézier, `Cglocal`, διαπιστώνουμε ότι

```
rank(Cglobal)=57
```

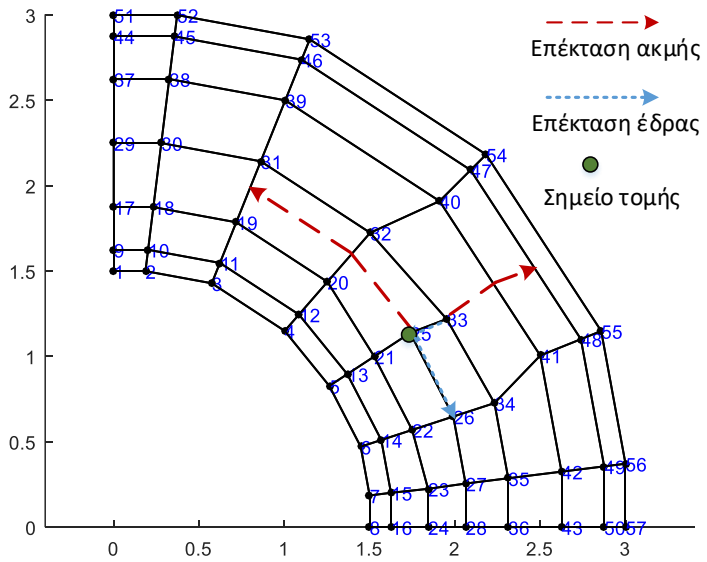
δηλαδή ότι η τάξη του καθολικού τελεστή είναι ίση με το πλήθος των δεσμών. Επομένως, σύμφωνα με τη σχέση (7.3) οι συναρτήσεις ανάμειξης T-spline είναι γραμμικά ανεξάρτητες και ορίζουν βάση για το χώρο T-spline και μπορούν να χρησιμοποιηθούν ως βάση για το χώρο λύσεων του προβλήματος συνοριακών τιμών που ορίζεται στο χωρίο T-spline.

Τέλος, όσον αφορά την ιδιότητα της διαμέρισης της μονάδας, το σύστημα (7.11) έχει λύση, η οποία δεν είναι το μοναδιαίο διάνυσμα, επομένως το πλέγμα είναι ημι-κανονικό. Συνεπώς, για την ικανοποίηση της αφινικής συνδιακύμανσης, η επίλυση του προβλήματος συνοριακών τιμών θα πρέπει να πραγματοποιηθεί χρησιμοποιώντας τις ρητές συναρτήσεις T-spline.

11.1.1 Η βάση T-spline

Έχοντας προσδιορίσει στο πλέγμα T-spline μια έγκυρη αλληλουχία κομβικών διαστημάτων, μπορούμε να ορίσουμε τις συναρτήσεις του T-spline. Γνωρίζουμε, γενικότερα, ότι σε κάθε δεσμό αντιστοιχεί από μια συνάρτηση, επομένως στα πλέγματα περιττού βαθμού (εν προκειμένω, τρίτου), όπου οι δεσμοί ταυτίζονται με τις κορυφές, κάθε συνάρτηση βάσης θα σχετίζεται με μια κορυφή.

Τα διανύσματα κομβικών διαστημάτων στο T-mesh διαθέτουν όλες τις πληροφορίες που χρειάζονται για την κατασκευή του παραμετρικού χώρο, εκτός από την τιμή του πρώτου κομβικού σημείου. Για παράδειγμα, για το διάνυσμα κομβικών διαστημάτων $\{1,3,2,1\}$ αν υποθέσουμε ότι ο πρώτος κόμβος έχει τιμή 0, το αντίστοιχο τοπικό κομβοδιάνυσμα είναι το $\{0,1,4,6,7\}$. Αντίστροφα, αν γνωρίζουμε το τοπικό κομβοδιάνυσμα, τότε το αντίστοιχο τοπικό διάνυσμα κομβικών διαστημάτων υπολογίζεται από τη διαφορά των γειτονικών κομβικών τιμών. Καθώς, στα T-spline δεν είναι απαραίτητη η γνώση του κόμβου αφετηρίας, είθισται να χρησιμοποιούνται τα διανύσματα κομβικών διαστημάτων. Παρ' όλα αυτά, για την επίτευξη συμβατότητας με τις μεθόδους σχεδιασμού του υπολογιστικού πακέτου, στην παρούσα εργασία, κρίνεται αναπόφευκτη η χρήση των ισοδύναμων τοπικών κομβοδιανυσμάτων.



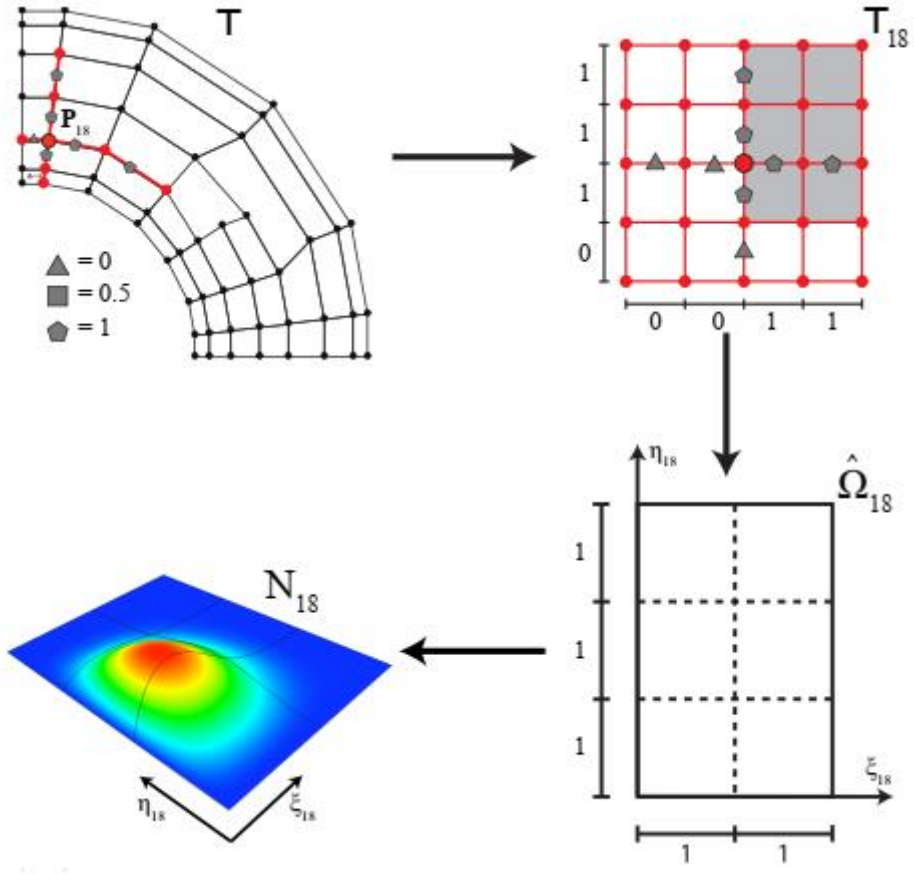
Σχήμα 11-4 Οι επεκτάσεις των διασταυρώσεων Τ T_{25} και T_{33}. Όπως γίνεται σαφές, οι επεκτάσεις τέμνονται στη διασταύρωση T_{25} μη κατάλληλο-προς-ανάλυση, σύμφωνα με τον ορισμό που δόθηκε στο Κεφάλαιο 4.3, παρ' όλα αυτά μπορεί η βάση T-spline μπορεί να χρησιμοποιηθεί για την εφαρμογή της Ισογεωμετρικής Ανάλυσης.

11.1.2 Η δομή του στοιχείου T-spline

Ένα στοιχείο T-spline $\Omega^e \subset \mathbb{R}^{d_s}$ είναι περιοχή στο φυσικό χώρο, η οποία φράσσεται από τις κομβικές γραμμές που είναι και γραμμές μειωμένης συνέχειας της βάσης T-spline. Οι συναρτήσεις βάσης στο εσωτερικό κάθε στοιχείου T-spline παρουσιάζουν συνέχεια C^∞.

Τα τοπικά πεδία $\widehat{\Omega}_{18}$ και $\widehat{\Omega}_{33}$ των συναρτήσεων βάσης N_{18} και N_{33}, αντίστοιχα, απεικονίζονται στην κάτω δεξιά γωνία των Σχημάτων Σχήμα 11-5 και Σχήμα 11-6, αντίστοιχα. Στην

περίπτωση του $\widehat{\Omega}_{18}$, έχουμε $\widehat{\Omega}_{18}^{1} = [0,2]$ και $\widehat{\Omega}_{18}^{2} = [0,3]$. Στην περίπτωση του $\widehat{\Omega}_{33}$, έχουμε $\widehat{\Omega}_{33}^{1} = [0,2]$ και $\widehat{\Omega}_{33}^{2} = [0,3]$.



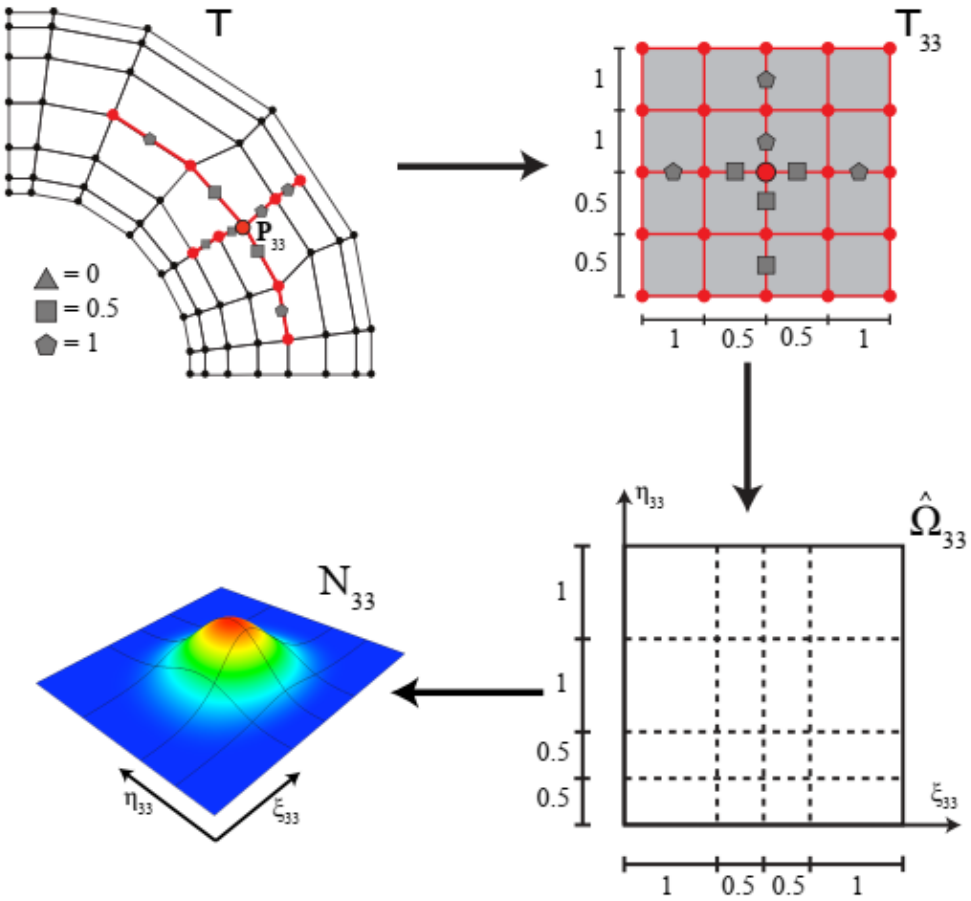
Σχήμα 11-5 Η κατασκευή της συνάρτησης βάσης T-spline N_{18} η οποία σχετίζεται με την κορυφή $\mathbf{P}_{18}$ του πλέγματος T-spline. Ξεκινώντας από πάνω αριστερά και συνεχίζοντας δεξιόστροφα: Η εξαγωγή των τοπικών κομβοδιανυσμάτων από το T-mesh, το τοπικό πλέγμα συνάρτησης βάσης που προκύπτει, το τοπικό πεδίο συνάρτησης βάσης και η συνάρτηση βάσης T-spline. [11]

11.1.2.1 Τοπικό πλέγμα συνάρτησης βάσης

Το τοπικό πλέγμα συνάρτησης βάσης, T_A ορίζεται ως η αναπαράσταση του πλέγματος τανυστικού γινομένου των τοπικών κομβοδιανυσμάτων, δηλαδή

$$T_A = \Xi_A^1 \otimes \Xi_A^2$$

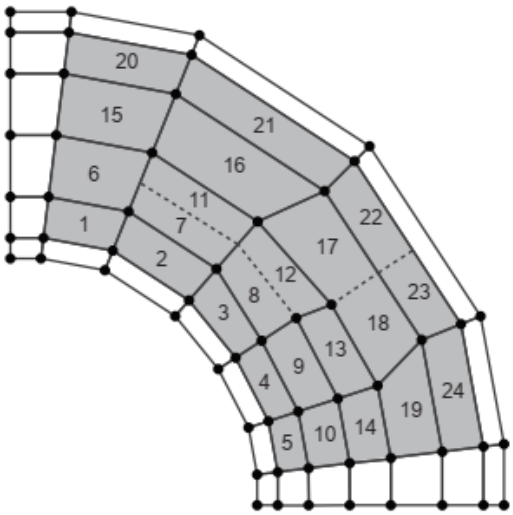
όπου Ξ_A^i τα τοπικά κομβοδιανύσματα της συνάρτησης N_A στην κατεύθυνση i. [11]



Σχήμα 11-6 Η κατασκευή της συνάρτησης βάσης T-spline N_{33} η οποία σχετίζεται με την κορυφη P_{33} του πλέγματος T-spline. Ξεκινώντας από πάνω αριστερά και συνεχίζοντας δεξιόστροφα: Η εξαγωγή των τοπικών κομβοδιανυσμάτων από το T-mesh, το τοπικό πλέγμα συνάρτησης βάσης που προκύπτει, το τοπικό πεδίο συνάρτησης βάσης και η συνάρτηση βάσης T-spline. [11]

11.1.2.2 Το στοιχειώδες T-mesh

Ένα στοιχείο T-mesh δεν έχει κατ' ανάγκην μία προς μία αντιστοιχία με ένα στοιχείο T-spline. Υπενθυμίζεται ότι ένα στοιχείο του πλέγματος T είναι ένα τετράπλευρο στο πλέγμα ελέγχου του T-spline ή στο T-mesh ενώ το στοιχείο T-spline είναι μια περιοχή της επιφάνειας T-spline που οριοθετείται από γραμμές μειωμένης συνέχειας στη βάση T-spline. Αυτό μπορεί να γίνει αντιληπτό σχεδιάζοντας το τοπικό πλέγμα T_{33} της συνάρτησης N_{33} πάνω στο πλέγμα T, όπως στο Σχήμα 11-6. Οι διακεκομμένες γραμμές στο Σχήμα 11-6 υποδεικνύουν ακμές που υπάρχουν στο T_{33} αλλά όχι στο T. Κάθε γραμμή κόμβου στη βάση T-spline είναι παρούσα σε τουλάχιστον μία συνάρτηση βάσης. Ωστόσο, δεν αντιπροσωπεύονται όλες αυτές οι γραμμές κόμβων από αντίστοιχες γραμμές στο πλέγμα T. Το στοιχειώδες T-mesh (elemental T-mesh) σχηματίζεται με την προσθήκη στο αρχικό T-mesh όλων των επιπλέον γραμμών μειωμένης συνέχειας. Το στοιχειώδες T-mesh του T παρουσιάζεται στο Σχήμα 11-7. Οι διακεκομμένες ακμές υποδεικνύουν τις γραμμές μειωμένης συνέχειας που δεν υπήρχαν στο αρχικό πλέγμα T.



Σχήμα 11-7 Τα στοιχεία T-spline στο στοιχειώδες T-mesh του T. [11]

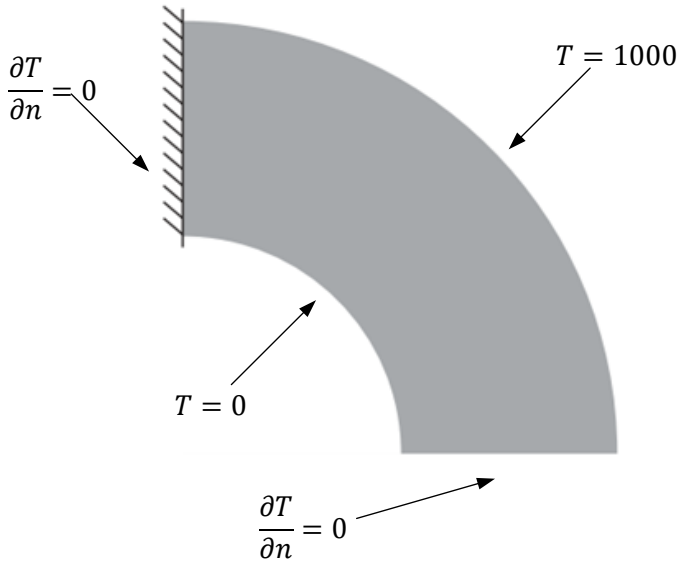
11.1.2.3 Μητρώο IEN

Σε κάθε στοιχείο αντιστοιχεί ένα σύνολο συναρτήσεων βάσης, οι οποίες εντός του στοιχείου είναι μη μηδενικές. Οι συναρτήσεις βάσης T-spline είναι σε μια-προς-μια αντιστοιχία με τα σημεία ελέγχου του πλέγματος T-spline και αριθμούνται σύμφωνα με την καθολική αρίθμηση των σημείων ελέγχου. Ο ρόλος του μητρώου IEN είναι να απεικονίζει τον τοπικό αριθμό της συνάρτησης βάσης a και τον αριθμό του στοιχείου e στον καθολικό αριθμό A του αντίστοιχου σημείου ελέγχου. Στην περίπτωση ενός T-spline με διασταυρώσεις T, το πλήθος των μη μηδενικών συναρτήσεων βάσης κάθε στοιχείου διαφοροποιείται σε κάθε στοιχείο, με $(p+1)^{d_p}$ να είναι το ελάχιστο πλήθος μη μηδενικών συναρτήσεων για καθένα στοιχείο, όπου d_p, το πλήθος των παραμετρικών συντεταγμένων. Αυτό έρχεται σε αντίθεση με τις συναρτήσεις NURBS, όπου όλα τα στοιχεία έχουν σταθερό πλήθος μη μηδενικών συναρτήσεων, ίσο με $(p+1)^{d_p}$. Εκτελώντας τη συνάρτηση υπολογισμού του μητρώου IEN, αποθηκεύεται στη μεταβλητή `IEN_array` ο πίνακας IEN για τη γεωμετρία T-spline της δοκού, όπως περιγράφεται στο script. Ο πίνακας `IEN_array` έχει διάσταση $\texttt{noelmnts} \times \max_{e}(\texttt{nbas}(e))$, όπου `noelmnts`, το πλήθος των στοιχείων του πλέγματος T-spline και $\texttt{nbas}(e)$, το πλήθος των συναρτήσεων βάσης που υποστηρίζονται στο στοιχείο e. Στο εν λόγω πλέγμα, το πλήθος των στοιχείων είναι 25 και όλα τα στοιχεία έχουν 16 μη μηδενικές συναρτήσεις βάσης, πλην των στοιχείων 9 και 10 που υποστηρίζουν 17 συναρτήσεις βάσης T-spline. Επομένως, το μητρώο IEN θα έχει διάσταση 25×17, με τη 17η στήλη να αποτελείται από το μηδενικό διάνυσμα, πλην των θέσεων 9 και 10. Το `IEN_array` παρουσιάζεται παρακάτω.

```
IEN_array =

   1    2    3    4    9   10   11   12   17   18   19   20   29   30   31   32    0

   2    3    4    5   10   11   12   13   18   19   20   21   25   30   31   32    0

   3    4    5    6   11   12   13   14   19   20   21   22   25   26   31   32    0

   4    5    6    7   12   13   14   15   20   21   22   23   25   26   27   32    0
```

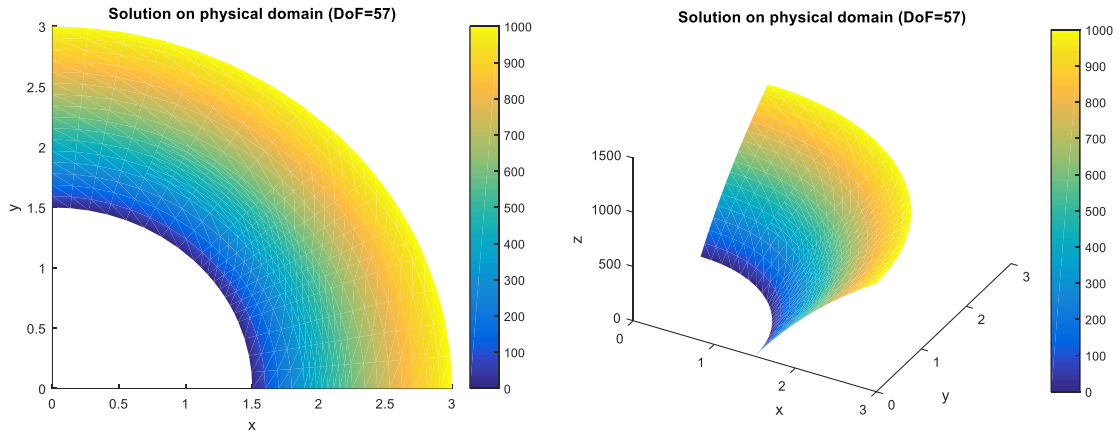
5	6	7	8	13	14	15	16	21	22	23	24	25	26	27	28	0
9	10	11	12	17	18	19	20	29	30	31	32	37	38	39	40	0
10	11	12	13	18	19	20	21	25	30	31	32	33	38	39	40	0
11	12	13	14	19	20	21	22	25	26	31	32	33	34	39	40	0
12	13	14	15	20	21	22	23	25	26	27	32	33	34	35	39	40
13	14	15	16	21	22	23	24	25	26	27	28	33	34	35	36	40
10	11	12	18	19	20	21	25	30	31	32	33	38	39	40	41	0
11	12	19	20	21	22	25	26	31	32	33	34	39	40	41	42	0
12	20	21	22	23	25	26	27	32	33	34	35	39	40	41	42	0
21	22	23	24	25	26	27	28	33	34	35	36	40	41	42	43	0
17	18	19	20	29	30	31	32	37	38	39	40	44	45	46	47	0
18	19	20	25	30	31	32	33	38	39	40	41	45	46	47	48	0
19	20	25	26	31	32	33	34	39	40	41	42	46	47	48	49	0
20	25	26	27	32	33	34	35	39	40	41	42	46	47	48	49	0
25	26	27	28	33	34	35	36	40	41	42	43	47	48	49	50	0
29	30	31	32	37	38	39	40	44	45	46	47	51	52	53	54	0
30	31	32	33	38	39	40	41	45	46	47	48	52	53	54	55	0
31	32	33	34	39	40	41	42	46	47	48	49	53	54	55	56	0
32	33	34	35	39	40	41	42	46	47	48	49	53	54	55	56	0
33	34	35	36	40	41	42	43	47	48	49	50	54	55	56	57	0

11.1.3 Διατύπωση προβλήματος συνοριακών τιμών



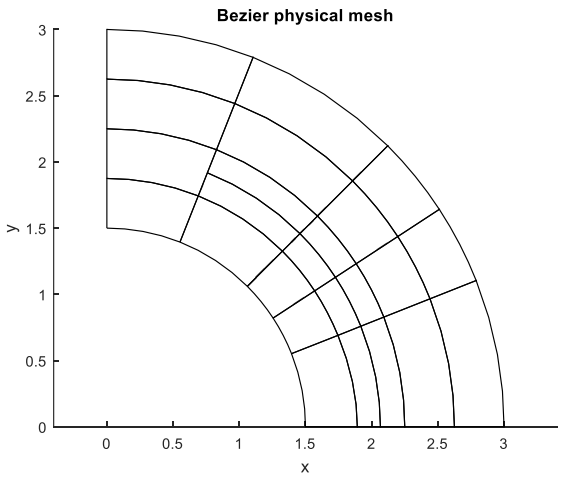
Σχήμα 11-8 Πρόβλημα συνοριακών τιμών Laplace με μεικτές συνοριακές συνθήκες ορισμένο στη δοκό τεταρτοκυκλίου

Το πρόβλημα Laplace, $\nabla^2 T = 0$, το οποίο ορίζεται πάνω στην προαναφερθείσα γεωμετρία έχει τις συνοριακές συνθήκες που απεικονίζονται στο Σχήμα 11-8. Στο δεξιό καμπύλο σύνορο εφαρμόζεται συνθήκη Dirichlet με $T = 1000$, ενώ στο αριστερό καμπύλο σύνορο η συνοριακή συνθήκη Dirichlet επιβάλλει η τιμή της θερμοκρασίας να είναι μηδενική ($T = 0$). Στα ευθύγραμμα συνοριακά τμήματα η συνοριακή συνθήκη είναι Neumann. Το εν λόγω πρόβλημα Laplace επιλύθηκε με συναρτήσεις T-spline με 57 βαθμούς ελευθερίας, συναρτήσεις NURBS με 64 βαθμούς ελευθερίας, ενώ πραγματοποιήθηκε και για τις δύο περιπτώσεις εξαγωγή Bézier. Επιπλέον, πέρα από την κλασική εξαγωγή Bézier, όπως έχει παρουσιαστεί σε πλήθος βιβλιογραφικών πηγών [21, 11, 6], η εξαγωγή Bézier υλοποιήθηκε στην πληρότητα της, δηλαδή, για την παρεμβολή της αριθμητικής λύσης χρησιμοποιήθηκαν τα ρητά πολυώνυμα Bernstein. Τέλος, οι παραπάνω υπολογιστικές προσεγγίσεις συγκρίθηκαν ως προς τη νόρμα L_2 του σφάλματος της αριθμητικής λύσης της καθεμίας.

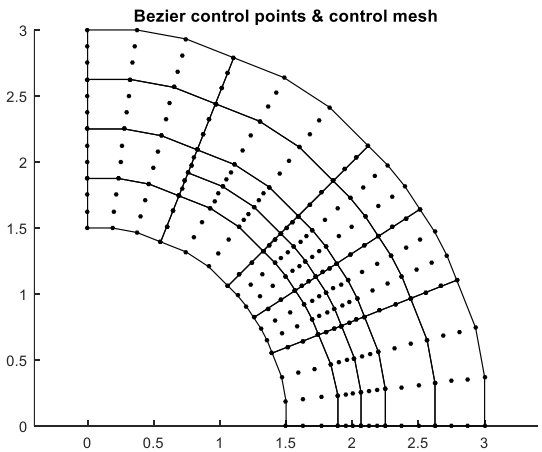


Σχήμα 11-9 Επίλυση με συναρτήσεις T-spline (DoF=57)

Στο Σχήμα 11-9 παρουσιάζεται το γράφημα της αριθμητικής λύσης (DoF=57), όπως ορίζεται στο φυσικό χωρίο. Η νόρμα L_2 του σφάλματος της λύσης είναι 0.0019%. Εκτελώντας την εξαγωγή Bézier στο πλέγμα T-spline, σχηματίζεται το φυσικό πλέγμα Bézier καθώς και τα νέα σημεία ελέγχου $\mathbf{P}^b$, τα οποία απεικονίζονται στο Σχήμα 11-10 και στο Σχήμα 11-11, αντίστοιχα.

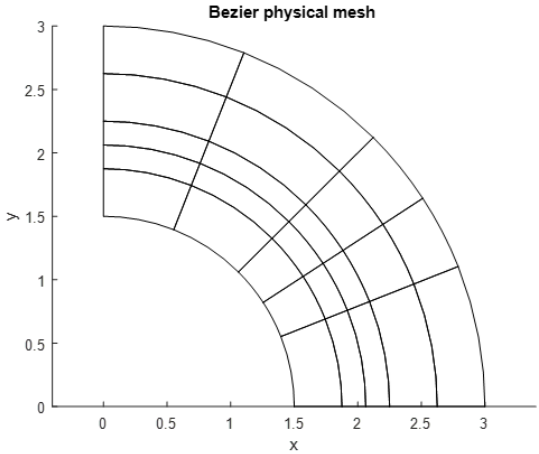


Σχήμα 11-10 Το φυσικό πλέγμα Bézier που σχηματίζεται από την Αποσύνθεση Bézier στο πλέγμα T-spline

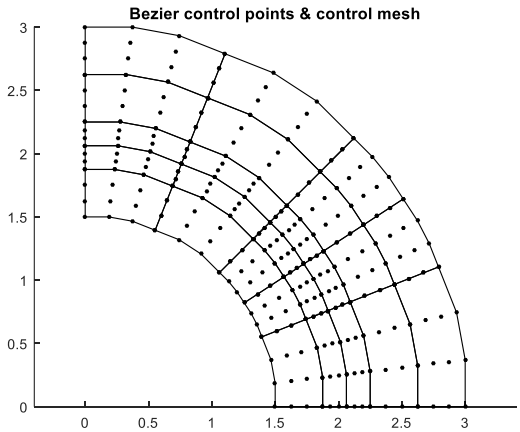


Σχήμα 11-11 Το πλέγμα ελέγχου Bézier με τα σημεία ελέγχου Bézier

Επιλύοντας το πρόβλημα Laplace με συναρτήσεις NURBS (DoF=64), η νόρμα L_2 του σφάλματος της αριθμητικής λύσης μειώνεται στο 0.0011%. Εκτελώντας την εξαγωγή Bézier στο πλέγμα NURBS σχηματίζεται το αντίστοιχο φυσικό πλέγμα Bézier και τα νέα σημεία ελέγχου Bézier, τα οποία απεικονίζονται στο Σχήμα 11-12 και στο Σχήμα 11-13, αντίστοιχα.



Σχήμα 11-12 Το φυσικό πλέγμα Bézier που σχηματίζεται από την Αποσύνθεση Bézier στο πλέγμα NURBS



Σχήμα 11-13 Το πλέγμα ελέγχου Bézier με τα σημεία ελέγχου Bézier, μετά την εφαρμογή της Αποσύνθεσης Bézier στο πλέγμα NURBS

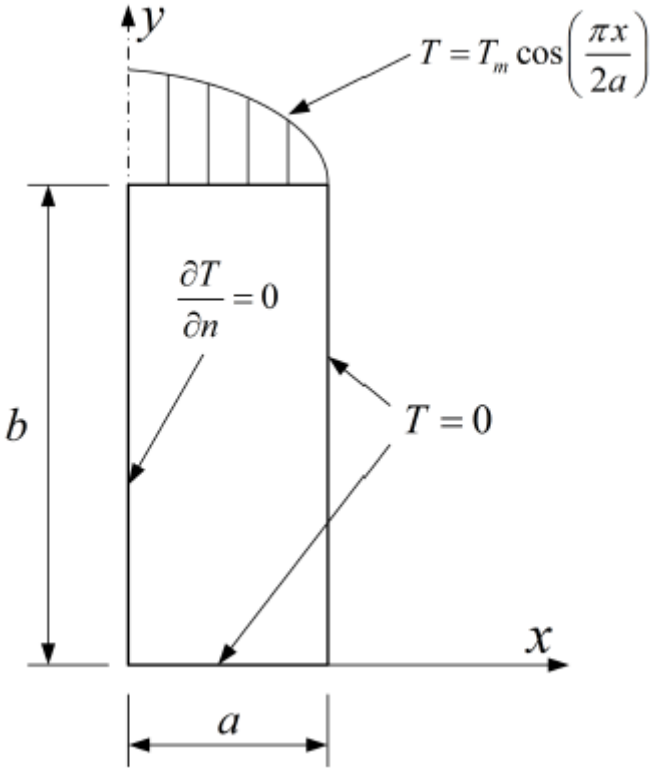
Τέλος, επιλύοντας το πρόβλημα Laplace στο πλέγμα NURBS, χρησιμοποιώντας τα ρητά πολυώνυμα Bernstein (DoF=256) που προκύπτουν από την Αποσύνθεση Bézier, παρατηρείται υποδιπλασιασμός της νόρμας L_2 του σφάλματος της λύσης (0.0005%). Δεδομένου ότι στην κλασική εφαρμογή της Αποσύνθεσης Bézier, απαιτείται η κατασκευή των τοπικών τελεστών εξαγωγής Bézier, ενώ εκτυπώνονται και τα τοπικά σημεία ελέγχου Bézier για κάθε στοιχείο, η ιδέα χρήσης των νέων συναρτήσεων Bernstein προκύπτει φυσικά. Η συναρμολόγηση του καθολικού μητρώου στιβαρότητας απαιτεί τη γνώση του νέου μητρώου IEN το οποίο λόγω της αυστηρής καρτεσιανής δομής του πλέγματος NURBS μπορεί να υπολογιστεί με ευκολία. Επομένως, ο υπολογιστικός κώδικας δεν επιβαρύνεται ιδιαίτερα με τη χρήση των πολυωνύμων Bernstein, παρ' ότι αυξάνεται το πλήθος των βαθμών ελευθερίας. Αξιοποιώντας, λοιπόν, την εκλέπτυνση του πλέγματος, επιτυγχάνεται επιπρόσθετη μείωση του σφάλματος, παρ' ότι χάνεται η ομαλότητα της λύσης που εξασφαλιζόταν από τις αρχικές συναρτήσεις NURBS. Φυσικά, η εν λόγω μέθοδος μπορεί να εφαρμοστεί και σε πλέγματα T-spline με διασταυρώσεις T. Η ιδιαιτερότητα όμως των πλεγμάτων T-spline δεν καθιστά τετριμμένη την κατασκευή του μητρώου IEN του εκλεπτυσμένου πλέγματος και απαιτείται κατάλληλος χειρισμός.

Στον παρακάτω πίνακα συγκεντρώνονται τα σφάλματα της κάθε μεθόδου ως προς την νόρμα L_2.

Νόρμα $\boldsymbol{L_2}$ σφάλματος αριθμητικής λύσης	Πλέγμα T-spline (DoF=$\mathbf{57}$)	Πλέγμα NURBS (DoF=$\mathbf{64}$)	Πλέγμα NURBS (με ρητά πολυώνυμα Bernstein, DoF=$\mathbf{256}$)
(%)	0.0019	0.0011	0.0005

11.2 Ορθογώνιο χωρίο $a \times b$

11.2.1 Διατύπωση προβλήματος συνοριακών τιμών



Σχήμα 11-14 Το ορθογώνιο χωρίο στο οποίο εφαρμόζεται το πρόβλημα Laplace με σημειωμένες τις συνοριακές συνθήκες. [24]

Στο [24] επιλύθηκε το πρόβλημα Laplace ($\nabla^2 T = 0$) σε ορθογώνιο χωρίο διάστασης $a \times b$ με μη ομοιογενείς συνοριακές συνθήκες Dirichlet και συνθήκες Neumann. Στο Σχήμα 11-14 εμφανίζονται με σαφήνεια οι συνοριακές συνθήκες στο χωρίο εφαρμογής του προβλήματος. Η σχεδίαση της γεωμετρίας και η επίλυση του προβλήματος έγινε με χρήση συναρτήσεων Bézier, B-spline και T-spline τρίτου βαθμού. Επιπλέον, πραγματοποιήθηκε εξαγωγή Bézier στις συναρτήσεις B-spline και η αριθμητική λύση υπολογίστηκε εκ νέου, αποδεικνύοντας την υπεροχή της εξαγωγής Bézier έναντι της συμβατικής Ισογεωμετρικής Ανάλυσης με B-spline, όσον αφορά την υπολογιστική ταχύτητα. Στην παρούσα εργασία θα επαναπροσεγγίσουμε αυτό το πρόβλημα, χρησιμοποιώντας το βελτιωμένο υπολογιστικό πακέτο του Laube. Αξίζει να σημειωθεί ότι η ολοκλήρωση πραγματοποιήθηκε στα στοιχεία όπως ορίζονται από τις γραμμές μειωμένης συνέχειας (T-spline elements), και στα οποία η τάξη συνέχειας των συναρτήσεων βάσης είναι σταθερή και όχι στα στοιχεία που ορίζονται από το πλέγμα T-spline (T-mesh elements), όπως πραγματοποιήθηκε στην εργασία [24]. Η παραπάνω λεπτομέρεια παρ' ότι δεν επιδρά σημαντικά στην ακρίβεια της αριθμητικής λύσης είναι σύμφωνη με τις υποθέσεις του κανόνα ολοκλήρωσης Gauss.

Η θεωρητική λύση του προβλήματος είναι

$$T(x,y) = T_m \left(\frac{\sinh\left(\frac{\pi y}{2a}\right)}{\sinh\left(\frac{\pi b}{2a}\right)} \right) \cos\left(\frac{\pi x}{2a}\right), \forall (x,y) \in \Omega = [0,a] \times [0,b]$$

όπου T_m η μέγιστη θερμοκρασία. Επιλέγουμε αυθαίρετα $\alpha = 3, b = 12$ και $T_m = 1000$.

Για την περιγραφή της γεωμετρίας και την προσέγγιση της αριθμητικής λύσης θα χρησιμοποιήσουμε ως συναρτήσεις βάσης:

α. Πολυώνυμα Bézier-Bernstein τρίτου βαθμού,
β. B-Spline τρίτου βαθμού με έναν και δυο εσωτερικούς κόμβους στα κομβοδιανύσματα και των δυο κατευθύνσεων,
γ. T-Spline τρίτου βαθμού, με T-mesh που προκύπτει από την εκλέπτυνση των άνω πλεγμάτων B-Spline στο άνω αριστερά κελί του εκάστοτε πλέγματος,
δ. B-Spline τρίτου βαθμού, τα οποία προκύπτουν από την μετατροπή των παραπάνω πλεγμάτων σε B-spline (T-spline to NURBS)
ε. T-spline τρίτου βαθμού, με επιπλέον τοπική εκλέπτυνση του παραπάνω πλέγματος B-spline
στ. B-Spline τρίτου βαθμού, τα οποία προκύπτουν από την μετατροπή του πλέγματος ε. σε B-spline (T-spline to NURBS)

Το πλέγμα της επιφάνειας Bézier απεικονίζεται στο Σχήμα 11-15. Για την κατασκευή της επιφάνειας, κατασκευάζουμε τον παραμετρικό χώρο, θεωρώντας το κομβοδιάνυσμα [0,0,0,0,3,3,3,3] για την κατεύθυνση ξ και [0,0,0,0,12,12,12,12] για την κατεύθυνση η. Επομένως, το πλέγμα τρίτου βαθμού αποτελείται από 16 δεσμούς. Επιπλέον, ορίζουμε ως σημεία ελέγχου της επιφάνειας τα σημεία (0,0,0),(1,0,0),(2,0,0), (3,0,0),(0,4,0), (1,4,0),(2,4,0),(3,4,0),(0,8,0),(1,8,0),(2,8,0),(3,8,0), (0,12,0),(1,12,0), (2,12,0),(3,12,0).

Το εν λόγω πλέγμα αποτέλεσε πλέγμα αναφοράς για τις εκλεπτύνσεις που πραγματοποιήθηκαν για την κατασκευή των επιφανειών B-spline με έναν και δύο εσωτερικούς κόμβους ανά κατεύθυνση.

Το σφάλμα της προσεγγιστικής λύσης με χρήση πολυωνύμων Bernstein είναι ίσο με 11.5574%.

Νόρμα $\boldsymbol{L_2}$ του σφάλματος	Bézier (DoF=**16**)
(%)	11.5574

i. B-spline με έναν εσωτερικό κόμβο ανά κατεύθυνση

Για την κατασκευή του πλέγματος B-spline με έναν εσωτερικό κόμβο ανά κατεύθυνση, εισάγεται ένας κόμβος στο μέσο των κομβοδιανυσμάτων και των δύο διαστάσεων. Δηλαδή, το νέο κομβοδιάνυσμα της κατεύθυνσης ξ είναι το $\left[0,0,0,0,\frac{3}{2},3,3,3,3\right]$ και για την κατεύθυνση η το

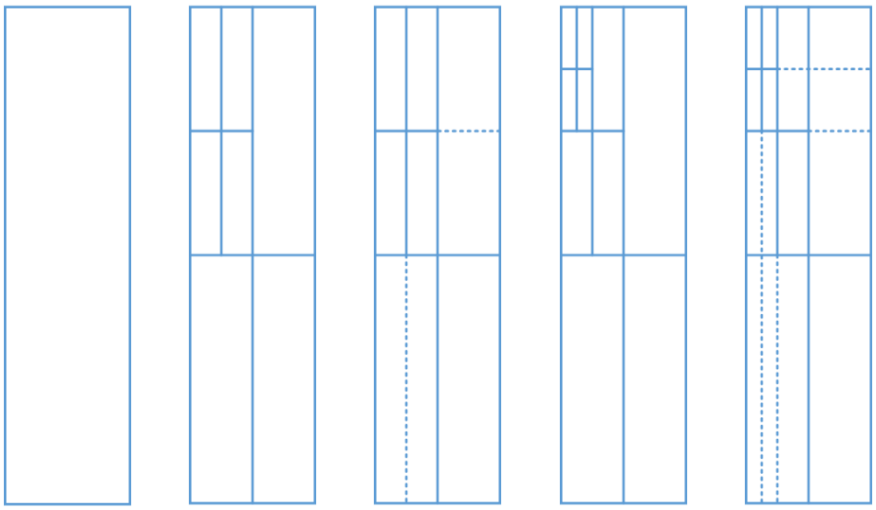
$\left[0,0,0,0,\frac{12}{2},12,12,12,12\right]$. Επομένως, σε αυτή την περίπτωση το πλέγμα B-spline θα αποτελείται από 25 δεσμούς.

Σχήμα 11-15 Πλέγμα αναφοράς Bézier

i. B-Spline με δύο εσωτερικούς κόμβους ομοιόμορφα κατανεμημένους ανά κατεύθυνση

Στο αρχικό πλέγμα Bézier, προσθέτουμε τις κομβικές τιμές $\frac{a}{3} = 1$ και $\frac{2a}{3} = 2$ στο αρχικό κομβοδιάνυσμα της κατεύθυνσης ξ και τις κομβικές τιμές $\frac{b}{3} = 4$ και $\frac{2b}{3} = 8$ στο αρχικό κομβοδιάνυσμα της κατεύθυνσης η. Συνολικά, το πλέγμα θα έχει 36 δεσμούς.

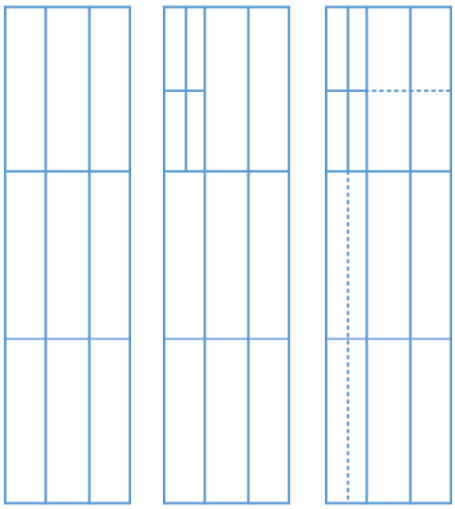
Στην περίπτωση που το αρχικό πλέγμα B-spline έχει έναν εσωτερικό κόμβο ανά κατεύθυνση, τα σφάλματα ανά πλέγμα εκλέπτυνσης παρουσιάζονται στον παρακάτω πίνακα:



DoF=25 DoF=32 DoF=36 DoF=39 DoF=**49**

Νόρμα L_2 σφάλματος	**B-spline**	**T-Spline** **μονή εκλέπτυνση**	**T-Spline 2 NURBS** **μονή εκλέπτυνση**	**T-Spline** **διπλή εκλέπτυνση**	**T-Spline 2 NURBS** **διπλή εκλέπτυνση**
%	3.7211	0.4119	0.3639	0.1842	0.1316

Ένω στην περίπτωση που έχουμε δύο ισαπέχοντες εσωτερικούς κόμβους ανά κατεύθυνση, παίρνουμε τα εξής σφάλματα για κάθε πλέγμα



	DoF=36	DoF=43	DoF=49
Νόρμα L_2 σφάλματος	**B-spline**	**T-Spline** **μία φορά εκλέπτυνση**	**T-Spline 2 NURBS** **μία φορά εκλέπτυνση**
(%)	1.0154	0.2613	0.2369

12 ΣΥΜΠΕΡΆΣΜΑΤΑ

Στην παρούσα εργασία διερευνήθηκε η χρήση ρητών συναρτήσεων T-spline τρίτου βαθμού στην Ισογεωμετρική Ανάλυση και εξετάστηκε η αποτελεσματικότητα της Αποσύνθεσης Bézier, ως εναλλακτικής προσέγγισης έναντι της συμβατικής Ισογεωμετρικής Ανάλυσης. Η υλοποίηση των μεθόδων έγινε στο ανοιχτό υπολογιστικό πρόγραμμα "T-splines for matlab" το οποίο για τους σκοπούς της παρούσας εργασίας, εμπλουτίστηκε και τροποποιήθηκε κατάλληλα. Η χρήση του εν λόγω εμπλουτισμένου πακέτου, αποτελεί μια απόπειρα δημιουργίας ενός αυτόνομου και ανοιχτού κώδικα, ο οποίος θα μπορεί να επιλύει βαθμωτά προβλήματα συνοριακών τιμών, όπως η εξίσωση θερμότητας, με τρόπο εύληπτο προς το χρήστη.

Από τα αποτελέσματα των υπολογιστικών εφαρμογών έγινε σαφές ότι η αύξηση των βαθμών ελευθερίας συνεπάγεται τη μείωση του σφάλματος των προσεγγιστικών λύσεων, όπως ήταν αναμενόμενο. Επιπλέον, η πρόταση της χρήσης των ρητών πολυωνύμων Bernstein του εκλεπτυσμένου πλέγματος NURBS επέδειξε έναν εναλλακτικό τρόπο υπολογισμού της αριθμητικής λύσης, ο οποίος αυξάνει την ακρίβεια, χωρίς την ιδιαίτερα μεγάλη επιβάρυνση του κώδικα με τον υπολογισμό επιπρόσθετων μεταβλητών. Καθώς στην κλασική εκδοχή της Αποσύνθεσης Bézier, ο πιο κοστοβόρος υπολογισμός είναι εκείνος των τελεστών εξαγωγής Bézier, η υλοποίηση της μεθόδου με ρητά πολυώνυμα Bernstein αποτελεί μια εύλογη επέκταση της κλασικής Αποσύνθεσης Bézier που ενώ μειώνει την τάξη συνέχειας της λύσης, αυξάνει αισθητά την ακρίβεια της.

13 ΜΕΛΛΟΝΤΙΚΉ ΈΡΕΥΝΑ

Η Ισογεωμετρική Ανάλυση αποτελεί μια ταχέως αναπτυσσόμενη προσέγγιση επίλυσης προβλημάτων μηχανικής φύσης, που προσελκύει την προσοχή όλο και περισσότερων ερευνητών των Εφαρμοσμένων Μαθηματικών καθώς και Μηχανικών. Ενδεικτικό χαρακτηριστικό αυτής της απήχησης είναι το γεγονός ότι στη μηχανή αναζήτησης Scholar της Google ο όρος "Isogeometric Analysis" απαριθμεί ήδη 20.000 αναφορές μέσα στα 17 χρόνια από την πρώτη παρουσίαση της μεθόδου με συναρτήσεις NURBS από τον Hughes [1]. Μέσα σε αυτά τα χρόνια, έχει πραγματοποιηθεί σημαντική πρόοδος και έχουν προκύψει πολλές ενδιαφέρουσες ιδέες που βελτιστοποιούν και εμπλουτίζουν την Ισογεωμετρική Ανάλυση. Μία από αυτές προτάθηκε από τους Bazilevs κ.ά. το 2010 [4] και αφορούσε τη χρήση συναρτήσεων T-spline έναντι των συναρτήσεων NURBS, η οποία προσέγγιση με τη σειρά της πυροδότησε μια αλληλουχία εργασιών και μελετών με θέμα την καταλληλότητα των T-spline στην Ισογεωμετρική Ανάλυση. Αντίστοιχα, μέσα από την Αποσύνθεση Bézier πραγματοποιήθηκε η γεφύρωση των διαφορών που χωρίζουν την Ισογεωμετρική Ανάλυση και την Ανάλυση των Πεπερασμένων Στοιχείων, δημιουργώντας επακολούθως το ερώτημα της υπεροχής ή μη στη χρήση των συναρτήσεων Bézier του εκλεπτυσμένου πλέγματος έναντι των αρχικών συναρτήσεων T-spline/NURBS στην επίλυση ενός βαθμωτού προβλήματος διαφορικών εξισώσεων. Αυτό το τελευταίο ερώτημα αποπειράθηκε να απαντήσει η παρούσα εργασία, συγκρίνοντας σε ένα πρόβλημα Laplace την ακρίβεια της λύσης με τις συναρτήσεις NURBS και με τις συναρτήσεις Bézier που προκύπτουν από την Αποσύνθεση Bézier στις αρχικές συναρτήσεις.

Η παραπάνω σύγκριση θα μπορούσε να πραγματοποιηθεί ανάμεσα σε συναρτήσεις T-spline και στις αντίστοιχες εκλεπτυσμένες συναρτήσεις Bézier. Σε αυτή την περίπτωση, θα έπρεπε να υπολογιστεί εκ νέου το μητρώο IEN του εκλεπτυσμένου πλέγματος. Καθώς η κατασκευή του αρχικού μητρώου IEN βασίστηκε στην αλγοριθμική κατασκευή του αρχικού γεωμετρικού μοντέλου T-spline, χρησιμοποιώντας την κλάση `tspline` του υπολογιστικού πακέτου, και δεδομένου ότι η εξαγωγή Bézier υλοποιήθηκε ανεξάρτητα από την αρχική τοπολογία και χωρίς εν τέλει να τη μετασχηματίζει, η κατασκευή του μητρώου IEN εντός αυτού του πλαισίου θα αύξανε την πολυπλοκότητα αυτής της προσέγγισης, ξεπερνώντας τους σκοπούς της παρούσας εργασίας. Αντ' αυτού, στην παρούσα εργασία η μέθοδος αυτή εφαρμόστηκε με επιτυχία στο πλέγμα NURBS, που λόγω της αυστηρά καρτεσιανής δομής του, η κατασκευή του νέου IEN προέκυψε φορμαλιστικά. Αποτελεί ένα ανοιχτό ερώτημα προς το/τη μελλοντικό/-η ερευνητή/-τρια η κατασκευή του μητρώου IEN για το εκλεπτυσμένο πλέγμα Bézier μιας τοπολογίας πλέγματος T-spline με διασταυρώσεις T. Επίσης, μια ενδιαφέρουσα παραλλαγή του κώδικα θα μπορούσε να επιλύσει προβλήματα ελαστικότητας ή ιδιοσυχνοτήτων επεκτείνοντας ακόμη περισσότερο το εύρος των δυνατοτήτων του υπολογιστικού πακέτου.

Επιπλέον, σημειώνεται ότι η παρούσα εργασία εστίασε σε διδιάστατα T-spline τρίτου βαθμού, χωρίς αυτό να υπονοεί ότι αντικείμενα T-spline περισσότερων διαστάσεων ή διαφορετικού βαθμού δεν αξίζουν προσοχής και διερεύνησης. Μάλιστα, σε πρόσφατη εργασία οι Morgenstern κ.ά. διεύρυναν την έννοια του κατάλληλου προς ανάλυση T-spline σε T-spline αυθαίρετου βαθμού και διάστασης. [17]

Τέλος, καθώς η Ισογεωμετρική Ανάλυση αναδεικνύεται ως μια ανερχόμενη μέθοδος επίλυσης διαφορικών εξισώσεων, έρχονται στο προσκήνιο νέες ιδέες που αφορούν τη βελτίωση της

εκλέπτυνσης των πλεγμάτων. Συνεπώς, προτείνονται εργαλεία βελτίωσης της σχεδιαστικής ακρίβειας, αλλά και, στο πλαίσιο της Ανάλυσης, μείωσης του σφάλματος της προσεγγιστικής λύσης. Ενδεικτική πρόταση είναι εκείνη των Li και Sederberg [25] που όρισαν ως μια γενίκευση των πλεγμάτων T-spline τα πλέγματα S-spline, τα οποία επιτρέπουν την εκλέπτυνση με ακόμη μικρότερο πλήθος περιττών σημείων ελέγχου. Όλες οι παραπάνω εξελίξεις μπορούν να δοκιμαστούν και σε αλγοριθμικό επίπεδο ανοίγοντας νέα πεδία έρευνας για το νέο/-α η ερευνητή/-τρια.

14 ΒΙΒΛΙΟΓΡΑΦΊΑ

[1] T. J. Hughes, J. A. Cottrell and Y. Bazilevs, “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement,” *Computer methods in applied mechanics and engineering,* vol. 194, no. 39-41, pp. 135-4195, 2005.

[2] J. A. Cottrell, T. J. Hughes and Y. Bazilevs, Isogeometric analysis: toward integration of CAD and FEA, John Wiley & Sons, 2009.

[3] C. G. Provatidis, Precursors of isogeometric analysis, Springer, 2019.

[4] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott and T. W. Sederberg, "Isogeometric analysis using T-splines," *Computer Methods in Applied Mechanics and Engineering,* vol. 199, no. 5-8, pp. 229-263, 2010.

[5] T. W. Sederberg, J. Zheng, A. Bakenov and A. Nasri, “T-splines and T-NURCCs,” *ACM transactions on graphics (TOG),* vol. 22, no. 3, pp. 477-484, 2003.

[6] T. N. Nguyen, “Isogeometric Finite Element Analysis based on Bézier Extraction of NURBS and T-Splines,” Norges teknisk-naturvitenskapelige universitet, 2011.

[7] 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),* vol. 23, no. 3, pp. 276-283, 2004.

[8] D. L. Cardon, “T-spline simplification,” Brigham Young University, 2007.

[9] J. Zhang and X. Li, “On the linear independence and partition of unity of arbitrary degree analysis-suitable T-splines,” *Communications in Mathematics and Statistics,* vol. 3, no. 3, pp. 353-364, 2015.

[10] J. Zhang and X. Li, “Local refinement for analysis-suitable++ T-splines,” *Computer Methods in Applied Mechanics and Engineering,* vol. 342, pp. 32-45, 2018.

[11] M. A. Scott, “T-splines as a design-through-analysis technology,” 2011.

[12] M. A. Scott, X. Li, T. W. Sederberg and T. J. Hughes, “Local refinement of analysis-suitable T-splines,” *Computer Methods in Applied Mechanics and Engineering,* vol. 213, pp. 206-222, 2012.

[13] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg and T. J. & Hughes, “Isogeometric finite element data structures based on Bézier extraction of T-splines,” *International Journal for Numerical Methods in Engineering,* vol. 88, no. 2, pp. 126-156, 2011.

[14] S. May, J. Vignollet and R. D. Borst, “The role of the Bézier extraction operator for T-splines of arbitrary degree: linear dependencies, partition of unity property, nesting behaviour and local refinement,” *International Journal for Numerical Methods in Engineering,* vol. 103, no. 8, pp. 547-581, 2015.

[15] X. Li, J. Zheng, T. W. Sederberg, T. J. Hughes and M. A. Scott, “On linear independence of T-spline blending functions,” *Computer Aided Geometric Design,* vol. 29, no. 1, pp. 63-76, 2012.

[16] A. Buffa and D. a. S. G. Cho, “Linear independence of the T-spline blending functions associated with some particular T-meshes,” *Computer Methods in Applied Mechanics and Engineering,* vol. 199, no. 23-24, pp. 1437-1445, 2010.

[17] P. Morgenstern and R. Görmer, "Analysis-Suitable T-Splines of arbitrary degree and dimension. 21: e202100234.," *Proc. Appl. Math. Mech.,* vol. 21, no. 1, 2021.

[18] M. J. Borden, M. A. Scott, J. A. Evans and T. J. Hughes, “Isogeometric finite element data structures based on Bézier extraction of NURBS,” *International Journal for Numerical Methods in Engineering,* vol. 87, no. 1-5, pp. 15-47, 2011.

[19] Χ. Προβατίδης, Η μέθοδος της Ισογεωμετρικής Ανάλυσης, Αθήνα: Εκδόσεις Κάλλιπος, 2022.

[20] T. J. Hughes, The finite element method: linear static and dynamic finite element analysis, 2000.

[21] O. J. Fredheim, “Isogeometric finite element analysis based on Bézier extraction of NURBS and T-splines,” Norges teknisk-naturvitenskapelige universitet, 2011.

[22] Ι. Δημητρίου, «TSPLINES -vs- NURBS (case study) v.1,» Αθήνα, 2020.

[23] L. Piegl and W. Tiller, The NURBS book, Springer Science & Business Media, 1996.

[24] Ι. Ραδίτσας, «Επίδοση της Αποσύζευξης Μπεζιέ στην Ισογεωμετρική Ανάλυση,» 2020.

[25] X. Li and T. W. Sederberg, “S-splines: A simple surface solution for IGA and CAD,” *Computer Methods in Applied Mechanics and Engineering*, vol. 350, pp. 664-678, 2019.

ΠΑΡΑΡΤΗΜΑ Α. ΚΩΔΙΚΕΣ

Α.1 Επιφάνεια Bézier τρίτου βαθμού

Α.1.1 Περιγραφή

Το παρακάτω πρόγραμμα αποτελεί εφαρμογή του παραδείγματος του Κεφαλαίου 10.1 και η εκτέλεση του βασίζεται στην αρχική δομή του υπολογιστικού πακέτου "t-splines for matlab". Αφορά την κατασκευή μιας επιφάνειας Bézier τρίτου βαθμού. Στον εν λόγω κώδικα ορίζονται οι δεσμοί/κορυφές του πλέγματος T-spline, η συνδεσιμότητα μεταξύ των κορυφών και τα σημεία ελέγχου της επιφάνειας. Τέλος, το πρόγραμμα σχεδιάζει τα σημεία της επιφάνειας Bézier.

Α.1.2 Κύριος Κώδικας

```
%Test1 T-Spline
%*****************************************%
%    ORIGINAL, by P. LAUBE (initially)      %
%*****************************************%
%-----------------------------------------------------------------
clear all; clc; close all;
%
p = 3;              % polynomial degree per direction
t = tSpline([],p);  % activate the main "subroutine"

%% BÉZIER p=3:
%{
  13oo14--------15oo16
   9oo10--------11oo12
    ||            ||
    ||            ||
    ||            ||
   5oo6----------7oo8
    oo------------oo
    1 2           3 4
%}

vertices = [ kVertex(0,0)       %anchor 1
             kVertex(0,0)       %anchor 2
             kVertex(1,0)       %anchor 3
             kVertex(1,0)       %anchor 4
             kVertex(0,0)       %anchor 5
             kVertex(0,0)       %anchor 6
             kVertex(1,0)       %anchor 7
             kVertex(1,0)       %anchor 8
             kVertex(0,1)       %anchor 9
             kVertex(0,1)       %anchor 10
             kVertex(1,1)       %anchor 11
             kVertex(1,1)       %anchor 12
             kVertex(0,1)       %anchor 13
             kVertex(0,1)       %anchor 14
```

```
              kVertex(1,1)        %anchor 15
              kVertex(1,1)];      %anchor 16

         t.kVertices=vertices;
         t.updateKnotVecs();
%Edges
%-----------------------------------
vertices(1).cnctRight(vertices(2));
vertices(1).cnctTop(vertices(5));
%---
vertices(2).cnctTop(vertices(6));
vertices(2).cnctRight(vertices(3));
vertices(2).cnctLeft(vertices(1));
%---
vertices(3).cnctTop(vertices(7));
vertices(3).cnctRight(vertices(4));
vertices(3).cnctLeft(vertices(2));
%---
vertices(4).cnctTop(vertices(8));
vertices(4).cnctLeft(vertices(3));
%---
vertices(5).cnctTop(vertices(9));
vertices(5).cnctRight(vertices(6));
vertices(5).cnctBottom(vertices(1));
%---
vertices(6).cnctTop(vertices(10));
vertices(6).cnctRight(vertices(7));
vertices(6).cnctBottom(vertices(2));
vertices(6).cnctLeft(vertices(5));

%---
vertices(7).cnctTop(vertices(11));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctBottom(vertices(3));
vertices(7).cnctLeft(vertices(6));
%---
vertices(8).cnctTop(vertices(12));
vertices(8).cnctBottom(vertices(4));
vertices(8).cnctLeft(vertices(7));
%---
vertices(9).cnctTop(vertices(13));
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(5));
%---
vertices(10).cnctTop(vertices(14));
vertices(10).cnctBottom(vertices(6));
vertices(10).cnctLeft(vertices(9));
vertices(10).cnctRight(vertices(11));
%---
vertices(11).cnctTop(vertices(15));
vertices(11).cnctRight(vertices(12));
```

```
vertices(11).cnctBottom(vertices(7));
vertices(11).cnctLeft(vertices(10));
%---
vertices(12).cnctTop(vertices(16));
vertices(12).cnctBottom(vertices(8));
vertices(12).cnctLeft(vertices(11));
%---
vertices(13).cnctRight(vertices(14));
vertices(13).cnctBottom(vertices(9));
%---
vertices(14).cnctRight(vertices(15));
vertices(14).cnctBottom(vertices(10));
vertices(14).cnctLeft(vertices(13));
%---
vertices(15).cnctRight(vertices(16));
vertices(15).cnctBottom(vertices(11));
vertices(15).cnctLeft(vertices(14));
%---
vertices(16).cnctLeft(vertices(15));
vertices(16).cnctBottom(vertices(12));
%---



t.kVertices = vertices;

%Control Points
%-----------------------------------
vertices(1).cPoint = [0  ;0  ;1];
vertices(2).cPoint = [1/3;0  ;1];
vertices(3).cPoint = [2/3;0  ;1];
vertices(4).cPoint = [1  ;0  ;1];

vertices(5).cPoint = [0  ;1/3  ;1];
vertices(6).cPoint = [1/3;1/3  ;1];
vertices(7).cPoint = [2/3;1/3  ;1];
vertices(8).cPoint = [1  ;1/3  ;1];

vertices(9).cPoint = [0  ;2/3  ;1];
vertices(10).cPoint = [1/3;2/3  ;1];
vertices(11).cPoint = [2/3;2/3  ;1];
vertices(12).cPoint = [1  ;2/3  ;1];

vertices(13).cPoint = [0  ;1  ;1];
vertices(14).cPoint = [1/3;1  ;1];
vertices(15).cPoint = [2/3;1  ;1];
vertices(16).cPoint = [1  ;1  ;1];

%-----------------------------------

% t.printPreImage();    %Color  Paper
```

```
% Give number of sample points to count the position per direction:
    sampleCount = 4;   %was 20
%-----------------------------------------------------------------
--------

% t.updateKnotVecs();
ps = linspace(0,1,sampleCount);
pt = linspace(0,1,sampleCount);

hold on;
for i=1:sampleCount
    for j=1:sampleCount
            P = t.evaluate(ps(i),pt(j));
            plot3(P(1),P(2),P(3),'ro');
        end
    end
end
```

A.2 Δοκός τεταρτοκυκλίου με T-spline

A.2.1 Περιγραφή

Στο παρακάτω πρόγραμμα κατασκευάζεται η επιφάνεια T-spline τρίτου βαθμού για μια δοκό σε σχήμα τεταρτοκυκλίου, σύμφωνα με το πλέγμα T-spline, το οποίο παρουσιάστηκε στο Κεφάλαιο 11.1. Η κατασκευή της γεωμετρίας βασίστηκε στο εμπλουτισμένο υπολογιστικό πακέτο. Επιπλέον, στην εν λόγω γεωμετρία επιλύθηκε το πρόβλημα Laplace με συνοριακές συνθήκες Dirichlet στα δύο καμπύλα σύνορα, το οποίο περιγράφεται στο Κεφάλαιο 11.1.3. Ο υπολογισμός της αριθμητικής λύσης πραγματοποιήθηκε ανάλυσης με τη χρήση των συναρτήσεων T-spline με 57 συνολικούς βαθμούς ελευθερίας σύμφωνα με τη μέθοδο της κλασικής ισογεωμετρικής και με τη μέθοδο της εξαγωγής Bézier. Όσον αφορά την πρώτη προσέγγιση, το μητρώο στιβαρότητας υπολογίστηκε με χρήση της συνάρτησης `blendfunNormW`, η οποία εξάγει τις συναρτήσεις T-spline και τις μερικές παραγώγους τους. Όσον αφορά τη μέθοδο της εξαγωγής Bézier, η επίλυση πραγματοποιήθηκε με δύο τρόπους. Στην πρώτη προσέγγιση, η συναρμολόγηση του μητρώου στιβαρότητας πραγματοποιήθηκε με χρήση της ρουτίνας `FormK_H`, εντός της οποίας υπολογίζονται τα μητρώα των τοπικών τελεστών εξαγωγής με βάση τη συνάρτηση `TsplineBasis`, η οποία υπολογίζει τις συναρτήσεις T-spline, αξιοποιώντας τους εν λόγω τοπικούς τελεστές. Η εν λόγω ρουτίνα βασίστηκε στον αλγόριθμο που παρουσιάστηκε στο [6] και ο οποίος στην αρχική του εκδοχή χειρίζεται αποκλειστικά συναρτήσεις NURBS. Στη δεύτερη προσέγγιση, η συναρμολόγηση του μητρώου στιβαρότητας πραγματοποιείται και πάλι με βοήθεια της συνάρτηση `TsplineBasis`, αλλά οι υπολογισμοί όλων των υπόλοιπων μεγεθών, όπως ο ιακωβιανός πίνακας, η ορίζουσα του καθώς και η συναρμολόγηση του μητρώου πραγματοποιούνται με ανεξάρτητο τρόπο. Τέλος, σχεδιάζεται το γράφημα των σημείων ελέγχου Bézier και το φυσικό πλέγμα, ενώ εκτυπώνεται το σφάλμα της προσεγγιστικής λύσης καθώς και οι τιμές της προσεγγιστικής και ακριβούς λύσης.

A.2.2 Κύριος Κώδικας

```
%% @SCOTT 2011 T-SPLINE
%%
%%  BÉZIER EXTRACTION (WITH WEIGHTS)

clear all;
t = tSpline([],3,[],[],[],[],[]);
% knots u                          knot v
% xi1=[0,0,1,1.5,2,3,4,4];       % et1=[0,0,1,1.5,2,3,4,4];
% xi2=[0,0,1,1.5,2,3,4,4];       % et2=[0,0,1,1.5,2,3,4,4];
% xi3=[0,0,1,1.5,2,3,4,4];       % et3=[0,0,1,1.5,2,3,4,4];
% xi4=[0,0,1,1.5,2];           % et4=[0,0,1,1.5];
% xi5=[0,0,1,2,3,4,4];       % et5=[0,0,1,1.5,2,3,4,4];
% xi6=[0,0,1,2,3,4,4];       % et6=[0,0,1,2,3,4,4];
% xi7=[0,0,1,2,3,4,4];       % et7=[0,0,1,2,3,4,4];
%xi8=[0,0,1,2,3,4,4];         % et8=[0,0,1,2,3,4,4];
%%-----------------
vertices = [ kVertex(0,0,0)      % anchor 1
    kVertex(0,0,0)      % anchor 2
    kVertex(1,0,0)      % anchor 3
    kVertex(2,0,0)      % anchor 4
    kVertex(2.5,0,0)    % anchor 5
    kVertex(3,0,0)      % anchor 6
    kVertex(4,0,0)      % anchor 7
    kVertex(4,0,0)      % anchor 8
    kVertex(0,0,0)      % anchor 9
    kVertex(0,0,0)      % anchor 10
    kVertex(1,0,0)      % anchor 11
    kVertex(2,0,0)      % anchor 12
    kVertex(2.5,0,0)    % anchor 13
    kVertex(3,0,0)      % anchor 14
    kVertex(4,0,0)      % anchor 15
    kVertex(4,0,0)      % anchor 16
    kVertex(0,1,0)      % anchor 17
    kVertex(0,1,0)      % anchor 18
    kVertex(1,1,0)      % anchor 19
    kVertex(2,1,0)      % anchor 20
    kVertex(2.5,1,0)    % anchor 21
    kVertex(3,1,0)      % anchor 22
    kVertex(4,1,0)      % anchor 23
    kVertex(4,1,0)      % anchor 24
    kVertex(2.5,1.5,0) % anchor 25
    kVertex(3,1.5,0)    % anchor 26
    kVertex(4,1.5,0)    % anchor 27
    kVertex(4,1.5,0)    % anchor 28
    kVertex(0,2,0)      % anchor 29
    kVertex(0,2,0)      % anchor 30
    kVertex(1,2,0)      % anchor 31
    kVertex(2,2,0)      % anchor 32
    kVertex(2.5,2,0)    % anchor 33
    kVertex(3,2,0)      % anchor 34
    kVertex(4,2,0)      % anchor 35
```

```
    kVertex(4,2,0)      % anchor 36
    kVertex(0,3,0)      % anchor 37
    kVertex(0,3,0)      % anchor 38
    kVertex(1,3,0)      % anchor 39
    kVertex(2,3,0)      % anchor 40
    kVertex(3,3,0)      % anchor 41
    kVertex(4,3,0)      % anchor 42
    kVertex(4,3,0)      % anchor 43
    kVertex(0,4,0)      % anchor 44
    kVertex(0,4,0)      % anchor 45
    kVertex(1,4,0)      % anchor 46
    kVertex(2,4,0)      % anchor 47
    kVertex(3,4,0)      % anchor 48
    kVertex(4,4,0)      % anchor 49
    kVertex(4,4,0)      % anchor 50
    kVertex(0,4,0)      % anchor 51
    kVertex(0,4,0)      % anchor 52
    kVertex(1,4,0)      % anchor 53
    kVertex(2,4,0)      % anchor 54
    kVertex(3,4,0)      % anchor 55
    kVertex(4,4,0)      % anchor 56
    kVertex(4,4,0)      % anchor 57
    ];


%% Edges/ Connectivity

vertices(1).cnctTop(vertices(9));
vertices(1).cnctRight(vertices(2));

vertices(2).cnctTop(vertices(10));
vertices(2).cnctRight(vertices(3));
vertices(2).cnctLeft(vertices(1));

vertices(3).cnctTop(vertices(11));
vertices(3).cnctRight(vertices(4));
vertices(3).cnctLeft(vertices(2));

vertices(4).cnctTop(vertices(12));
vertices(4).cnctRight(vertices(5));
vertices(4).cnctLeft(vertices(3));

vertices(5).cnctTop(vertices(13));
vertices(5).cnctRight(vertices(6));
vertices(5).cnctLeft(vertices(4));

vertices(6).cnctTop(vertices(14));
vertices(6).cnctRight(vertices(7));
vertices(6).cnctLeft(vertices(5));
```

```
vertices(7).cnctTop(vertices(15));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctLeft(vertices(6));

vertices(8).cnctTop(vertices(16));
vertices(8).cnctLeft(vertices(7));

vertices(9).cnctTop(vertices(17));
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(1));

vertices(10).cnctTop(vertices(18));
vertices(10).cnctRight(vertices(11));
vertices(10).cnctBottom(vertices(2));
vertices(10).cnctLeft(vertices(9));

vertices(11).cnctTop(vertices(19));
vertices(11).cnctRight(vertices(12));
vertices(11).cnctBottom(vertices(3));
vertices(11).cnctLeft(vertices(10));

vertices(12).cnctTop(vertices(20));
vertices(12).cnctRight(vertices(13));
vertices(12).cnctBottom(vertices(4));
vertices(12).cnctLeft(vertices(11));

vertices(13).cnctTop(vertices(21));
vertices(13).cnctRight(vertices(14));
vertices(13).cnctBottom(vertices(5));
vertices(13).cnctLeft(vertices(12));

vertices(14).cnctTop(vertices(22));
vertices(14).cnctRight(vertices(15));
vertices(14).cnctBottom(vertices(6));
vertices(14).cnctLeft(vertices(13));

vertices(15).cnctTop(vertices(23));
vertices(15).cnctRight(vertices(16));
vertices(15).cnctBottom(vertices(7));
vertices(15).cnctLeft(vertices(14));

vertices(16).cnctTop(vertices(24));
vertices(16).cnctBottom(vertices(8));
vertices(16).cnctLeft(vertices(15));

vertices(17).cnctTop(vertices(29));
vertices(17).cnctRight(vertices(18));
vertices(17).cnctBottom(vertices(9));

vertices(18).cnctTop(vertices(30));
```

```
vertices(18).cnctRight(vertices(19));
vertices(18).cnctBottom(vertices(10));
vertices(18).cnctLeft(vertices(17));

vertices(19).cnctTop(vertices(31));
vertices(19).cnctRight(vertices(20));
vertices(19).cnctBottom(vertices(11));
vertices(19).cnctLeft(vertices(18));

vertices(20).cnctTop(vertices(32));
vertices(20).cnctRight(vertices(21));
vertices(20).cnctBottom(vertices(12));
vertices(20).cnctLeft(vertices(19));

vertices(21).cnctTop(vertices(25));
vertices(21).cnctRight(vertices(22));
vertices(21).cnctBottom(vertices(13));
vertices(21).cnctLeft(vertices(20));

vertices(22).cnctTop(vertices(26));
vertices(22).cnctRight(vertices(23));
vertices(22).cnctBottom(vertices(14));
vertices(22).cnctLeft(vertices(21));

vertices(23).cnctTop(vertices(27));
vertices(23).cnctRight(vertices(24));
vertices(23).cnctBottom(vertices(15));
vertices(23).cnctLeft(vertices(22));

vertices(24).cnctTop(vertices(28));
vertices(24).cnctBottom(vertices(16));
vertices(24).cnctLeft(vertices(23));

vertices(25).cnctTop(vertices(33));
vertices(25).cnctRight(vertices(26));
vertices(25).cnctBottom(vertices(21));

vertices(26).cnctTop(vertices(34));
vertices(26).cnctRight(vertices(27));
vertices(26).cnctBottom(vertices(22));
vertices(26).cnctLeft(vertices(25));

vertices(27).cnctTop(vertices(35));
vertices(27).cnctRight(vertices(28));
vertices(27).cnctBottom(vertices(23));
vertices(27).cnctLeft(vertices(26));

vertices(28).cnctTop(vertices(36));
vertices(28).cnctBottom(vertices(24));
```

```
vertices(28).cnctLeft(vertices(27));

vertices(29).cnctTop(vertices(37));
vertices(29).cnctRight(vertices(30));
vertices(29).cnctBottom(vertices(17));

vertices(30).cnctTop(vertices(38));
vertices(30).cnctRight(vertices(31));
vertices(30).cnctBottom(vertices(18));
vertices(30).cnctLeft(vertices(29));

vertices(31).cnctTop(vertices(39));
vertices(31).cnctRight(vertices(32));
vertices(31).cnctBottom(vertices(19));
vertices(31).cnctLeft(vertices(30));

vertices(32).cnctTop(vertices(40));
vertices(32).cnctRight(vertices(33));
vertices(32).cnctBottom(vertices(20));
vertices(32).cnctLeft(vertices(31));

vertices(33).cnctRight(vertices(34));
vertices(33).cnctBottom(vertices(25));
vertices(33).cnctLeft(vertices(32));

vertices(34).cnctTop(vertices(41));
vertices(34).cnctRight(vertices(35));
vertices(34).cnctBottom(vertices(26));
vertices(34).cnctLeft(vertices(33));

vertices(35).cnctTop(vertices(42));
vertices(35).cnctRight(vertices(36));
vertices(35).cnctBottom(vertices(27));
vertices(35).cnctLeft(vertices(34));

vertices(36).cnctTop(vertices(43));
vertices(36).cnctBottom(vertices(28));
vertices(36).cnctLeft(vertices(35));

vertices(37).cnctTop(vertices(44));
vertices(37).cnctRight(vertices(38));
vertices(37).cnctBottom(vertices(29));

vertices(38).cnctTop(vertices(45));
vertices(38).cnctRight(vertices(39));
vertices(38).cnctBottom(vertices(30));
vertices(38).cnctLeft(vertices(37));

vertices(39).cnctTop(vertices(46));
vertices(39).cnctRight(vertices(40));
```

```
vertices(39).cnctBottom(vertices(31));
vertices(39).cnctLeft(vertices(38));

vertices(40).cnctTop(vertices(47));
vertices(40).cnctRight(vertices(41));
vertices(40).cnctBottom(vertices(32));
vertices(40).cnctLeft(vertices(39));

vertices(41).cnctTop(vertices(48));
vertices(41).cnctRight(vertices(42));
vertices(41).cnctBottom(vertices(34));
vertices(41).cnctLeft(vertices(40));

vertices(42).cnctTop(vertices(49));
vertices(42).cnctRight(vertices(43));
vertices(42).cnctBottom(vertices(35));
vertices(42).cnctLeft(vertices(41));

vertices(43).cnctTop(vertices(50));
vertices(43).cnctBottom(vertices(36));
vertices(43).cnctLeft(vertices(42));

vertices(44).cnctTop(vertices(51));
vertices(44).cnctRight(vertices(45));
vertices(44).cnctBottom(vertices(37));

vertices(45).cnctTop(vertices(52));
vertices(45).cnctRight(vertices(46));
vertices(45).cnctBottom(vertices(38));
vertices(45).cnctLeft(vertices(44));

vertices(46).cnctTop(vertices(53));
vertices(46).cnctRight(vertices(47));
vertices(46).cnctBottom(vertices(39));
vertices(46).cnctLeft(vertices(45));

vertices(47).cnctTop(vertices(54));
vertices(47).cnctRight(vertices(48));
vertices(47).cnctBottom(vertices(40));
vertices(47).cnctLeft(vertices(46));

vertices(48).cnctTop(vertices(55));
vertices(48).cnctRight(vertices(49));
vertices(48).cnctBottom(vertices(41));
vertices(48).cnctLeft(vertices(47));

vertices(49).cnctTop(vertices(56));
vertices(49).cnctRight(vertices(50));
vertices(49).cnctBottom(vertices(42));
```

```
vertices(49).cnctLeft(vertices(48));

vertices(50).cnctTop(vertices(57));
vertices(50).cnctBottom(vertices(43));
vertices(50).cnctLeft(vertices(49));

vertices(51).cnctRight(vertices(52));
vertices(51).cnctBottom(vertices(44));

vertices(52).cnctRight(vertices(53));
vertices(52).cnctBottom(vertices(45));
vertices(52).cnctLeft(vertices(51));

vertices(53).cnctRight(vertices(54));
vertices(53).cnctBottom(vertices(46));
vertices(53).cnctLeft(vertices(52));

vertices(54).cnctRight(vertices(55));
vertices(54).cnctBottom(vertices(47));
vertices(54).cnctLeft(vertices(53));

vertices(55).cnctRight(vertices(56));
vertices(55).cnctBottom(vertices(48));
vertices(55).cnctLeft(vertices(54));

vertices(56).cnctRight(vertices(57));
vertices(56).cnctBottom(vertices(49));
vertices(56).cnctLeft(vertices(55));

vertices(57).cnctBottom(vertices(50));
vertices(57).cnctLeft(vertices(56));

%-----------
t.kVertices = vertices;

t.updateKnotVecs();

% t.printPreImagePaper();

% U_vec, V_vec

numberOfVertices = size(vertices,1);
ncolumns=5;
for i=1:numberOfVertices
    U_vec(i,1:ncolumns)=vertices(i,1).knotVec(1,:);  %ncolumns=5, for p
= 3.
    V_vec(i,1:ncolumns)=vertices(i,1).knotVec(2,:);  %ncolumns=5, for p
= 3.
end
```

```
U_vec;
V_vec;



% figure
% t.printPreImagePaper();

%% CONTROL POINTS

vertices(1).cPoint = [0;1.5;0];
vertices(2).cPoint = [0.1858;1.5;0];
vertices(3).cPoint = [0.5746;1.4288;0];
vertices(4).cPoint = [1.0022;1.1497;0];
vertices(5).cPoint = [1.2637;0.8275;0];
vertices(6).cPoint = [1.4477;0.4714;0];
vertices(7).cPoint = [1.5;0.1858;0];
vertices(8).cPoint = [1.5;0;0];
vertices(9).cPoint = [0;1.625;0];
vertices(10).cPoint = [0.2013;1.625;0];
vertices(11).cPoint = [0.6224;1.5479;0];
vertices(12).cPoint = [1.0857;1.2455;0];
vertices(13).cPoint = [1.3690;0.8964;0];
vertices(14).cPoint = [1.5683;0.5107;0];
vertices(15).cPoint = [1.6250;0.2013;0];
vertices(16).cPoint = [1.6250;0;0];
vertices(17).cPoint = [0;1.8750;0];
vertices(18).cPoint = [0.2323;1.8750;0];
vertices(19).cPoint = [0.7182;1.7860;0];
vertices(20).cPoint = [1.2527;1.4371;0];
vertices(21).cPoint = [1.5270;0.9999;0];
vertices(22).cPoint = [1.7493;0.5696;0];
vertices(23).cPoint = [1.8425;0.2246;0];
vertices(24).cPoint = [1.8425;0;0];
vertices(25).cPoint = [1.7376;1.1378;0];
vertices(26).cPoint = [1.9906;0.6482;0];
vertices(27).cPoint = [2.0625;0.2555;0];
vertices(28).cPoint = [2.0625;0;0];
vertices(29).cPoint = [0;2.2500;0];
vertices(30).cPoint = [0.2788;2.2500;0];
vertices(31).cPoint = [0.8618;2.1432;0];
vertices(32).cPoint = [1.5033;1.7245;0];
vertices(33).cPoint = [1.9516;1.2194;0];
vertices(34).cPoint = [2.2319;0.7268;0];
vertices(35).cPoint = [2.3125;0.2865;0];
vertices(36).cPoint = [2.3125;0;0];
vertices(37).cPoint = [0;2.6250;0];
vertices(38).cPoint = [0.3252;2.6250;0];
vertices(39).cPoint = [1.0055;2.5004;0];
vertices(40).cPoint = [1.9100;1.9100;0];
vertices(41).cPoint = [2.5004;1.0055;0];
vertices(42).cPoint = [2.6250;0.3252;0];
```

```
vertices(43).cPoint = [2.6250;0;0];
vertices(44).cPoint = [0;2.8750;0];
vertices(45).cPoint = [0.3562;2.8750;0];
vertices(46).cPoint = [1.1012;2.7386;0];
vertices(47).cPoint = [2.0919;2.0919;0];
vertices(48).cPoint = [2.7386;1.1012;0];
vertices(49).cPoint = [2.8750;0.3562;0];
vertices(50).cPoint = [2.8750;0;0];
vertices(51).cPoint = [0;3;0];
vertices(52).cPoint = [0.3717;3;0];
vertices(53).cPoint = [1.1491;2.8576;0];
vertices(54).cPoint = [2.1829;2.1829;0];
vertices(55).cPoint = [2.8576;1.1491;0];
vertices(56).cPoint = [3;0.3717;0];
vertices(57).cPoint = [3;0;0];

%weights==1
for i=1:numberOfVertices
    vertices(i).weight=1;
end

%weights~=1
% %{
vertices(1).weight=1;
vertices(2).weight=0.9512;
vertices(3).weight=0.8780;
vertices(4).weight=0.8475;
vertices(5).weight=0.8597;
vertices(6).weight=0.8963;
vertices(7).weight=0.9512;
vertices(8).weight=1;
vertices(9).weight=1;
vertices(10).weight=0.9512;
vertices(11).weight=0.8780;
vertices(12).weight=0.8475;
vertices(13).weight=0.8597;
vertices(14).weight=0.8963;
vertices(15).weight=0.9512;
vertices(16).weight=1;
vertices(17).weight=1;
vertices(18).weight=0.9512;
vertices(19).weight=0.8780;
vertices(20).weight=0.8475;
vertices(21).weight=0.8597;
vertices(22).weight=0.8963;
vertices(23).weight=0.9512;
vertices(24).weight=1;
vertices(25).weight=0.8597;
vertices(26).weight=0.8963;
vertices(27).weight=0.9512;
vertices(28).weight=1;
vertices(29).weight=1;
```

```
vertices(30).weight=0.9512;
vertices(31).weight=0.8780;
vertices(32).weight=0.8475;
vertices(33).weight=0.7895;
vertices(34).weight=0.8963;
vertices(35).weight=0.9512;
vertices(36).weight=1;
vertices(37).weight=1;
vertices(38).weight=0.9512;
vertices(39).weight=0.8780;
vertices(40).weight=0.8413;
vertices(41).weight=0.8780;
vertices(42).weight=0.9512;
vertices(43).weight=1;
vertices(44).weight=1;
vertices(45).weight=0.9512;
vertices(46).weight=0.8780;
vertices(47).weight=0.8413;
vertices(48).weight=0.8780;
vertices(49).weight=0.9512;
vertices(50).weight=1;
vertices(51).weight=1;
vertices(52).weight=0.9512;
vertices(53).weight=0.8780;
vertices(54).weight=0.8413;
vertices(55).weight=0.8780;
vertices(56).weight=0.9512;
vertices(57).weight=1;

%}

t.kVertices=vertices;

p=3; nanchors=57;
% return
%% Ctrl Points coordinates

for i=1:nanchors
    XYctrl(1,i)=t.kVertices(i).cPoint(1);
    XYctrl(2,i)=t.kVertices(i).cPoint(2);
    XYctrl(3,i)=t.kVertices(i).cPoint(3);
end



t.printcontrolmesh;

%{
%% T-Spline Evaluation
sampleCount=20;
```

```
ps = linspace(0,4,sampleCount);
pt = linspace(0,4,sampleCount);
pr = linspace(0,4,sampleCount);

%----------- PATCHES
nopatches=42;
patches=patches_calc(t,nopatches);

%----------- ELEMENTS

t.fVert_last
fVertices=t.fVertices;

fVertices(1).cnctTop(vertices(32));
vertices(32).cnctBottom(fVertices(1));
fVertices(1).cnctBottom(vertices(20));
vertices(20).cnctTop(fVertices(1));

fVertices(2).cnctTop(vertices(31));
vertices(31).cnctBottom(fVertices(2));
fVertices(2).cnctBottom(vertices(19));
vertices(19).cnctTop(fVertices(2));

fVertices(3).cnctRight(vertices(41));
vertices(41).cnctLeft(fVertices(3));
fVertices(3).cnctLeft(vertices(40));
vertices(40).cnctRight(fVertices(3));

fVertices(4).cnctRight(vertices(48));
vertices(48).cnctLeft(fVertices(4));
fVertices(4).cnctLeft(vertices(47));
vertices(47).cnctRight(fVertices(4));

fVertices=t.fVertices;
%%
%%Compute the elements
noelmnts=24;
elmnts_pts=element_calc(t,noelmnts);
%%
% Gauss points and weights:
[gi,ome] = pgau_rectangle_FULL;
ngaussx=p+1; %2*p   %ceil((1+p^2)/2);
ngaussy=ngaussx;

%% Stiffness matrix
K=zeros(nanchors);  %initialize stiffness matrix
M=zeros(nanchors);  %initialize stiffness matrix
area=0;
for i=1:nanchors
    weights(i)=t.kVertices(i).weight;
```

```
end
IEN_array=IEN_calc(vertices,U_vec,V_vec,elmnts_pts,nanchors);
nanchorsEl=zeros(noelmnts,1);
% return
nanchorsEl(:)=sum(IEN_array~=0,2);
for k=1:noelmnts
    nbas = nanchorsEl(k);
    ianchory=zeros(1,nbas);

    ianchory(1:nbas)=nonzeros(IEN_array(k,:));

    %Corner points:
    xA=elmnts_pts(k,1); yA=elmnts_pts(k,2);
    xB=elmnts_pts(k,3); yB=elmnts_pts(k,4);
    xC=elmnts_pts(k,5); yC=elmnts_pts(k,6);
    xD=elmnts_pts(k,7); yD=elmnts_pts(k,8);
    %Centroid:
    xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
    %Jacobian:
    detJ  =  (xB-xA)*(yC-yB)/4;  %determinant  of  the  mapping  to  the
parameter space
    %
    if(detJ ~= 0)
        taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx); %mapping gaussian
points to the parameter space
        taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);

        Bipmat_dy=zeros(nbas,ngaussy,ngaussx);
        Bipmat_dx=zeros(nbas,ngaussy,ngaussx);
        Bipmat=zeros(nbas,ngaussy,ngaussx);

        %
        for igx=1:ngaussy
            for igy=1:ngaussx
                [~,                ~,Bip,Bip_dx,Bip_dy]                =
t.blendfunNormW(ianchory,taugaussx(igy), taugaussy(igx));
                Bipmat_dy(:,igx,igy)=Bip_dy(:);
                Bipmat(:,igx,igy)=Bip(:);
                Bipmat_dx(:,igx,igy)=Bip_dx(:);
            end
        end

        Nip;
        dNip_dx;

        XYctrloc=XYctrl(:,ianchory);

        shp=zeros(3,nbas);

        for igx=1:ngaussy
```

```
            for igy=1:ngaussx

                %-----CONSTRUCT JACOBIAN AND ITS INVERSE
                %                    nel = nanchorsEl(k);     %number of
control points in the T-mesh

                shp(3,1:nbas)=Bipmat(1:nbas,igx,igy);
                shp(1,1:nbas)=Bipmat_dx(1:nbas,igx,igy);
                shp(2,1:nbas)=Bipmat_dy(1:nbas,igx,igy);
                for ii=1:2
                    for jj=1:2
                        xs(ii,jj)=0;
                        for m=1:nbas

xs(ii,jj)=xs(ii,jj)+XYctrloc(ii,m)*shp(jj,m);    %Jacobian <------
                        end
                    end
                end

                %---
                xsj=xs(1,1)*xs(2,2)-xs(1,2)*xs(2,1) ; %detJ
                sx(1,1)=xs(2,2)/xsj ; %Inverse of Jacobian
                sx(2,2)=xs(1,1)/xsj ;
                sx(1,2)=-xs(1,2)/xsj;
                sx(2,1)=-xs(2,1)/xsj;
                %-----FORM GLOBAL DERIVATIVES
                for ii=1:nbas
                    tp       = shp(1,ii)*sx(1,1)+shp(2,ii)*sx(2,1);
                    shp(2,ii)   =   shp(1,ii)*sx(1,2)+shp(2,ii)*sx(2,2);
%y-derivative
                    shp(1,ii)=tp ;                                  %x-
derivative
                end

                shp(1,:);
                %---Store detJ:
                DETJacob(igx,igy)=xsj;

                %              DETJacob


                area               =               area               +
DETJacob(igx,igy)*detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                fprintf('Element=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

                for idof=1:nbas
                    ii=IEN_array(k,idof);

                    dNix=shp(1,idof);
```

```
                    dNiy=shp(2,idof);
                    for jdof=1:nbas
                        jj=IEN_array(k,jdof);
                        dNjx=shp(1,jdof);
                        dNjy=shp(2,jdof);

                        K(ii,jj)=K(ii,jj)+(dNix*dNjx+dNiy*dNjy)*...

DETJacob(igx,igy)*detJ*ome(igy,ngaussx)*ome(igx,ngaussy);

                        M(ii,jj)=M(ii,jj)+shp(3,idof)*shp(3,jdof)*...

DETJacob(igx,igy)*detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                    end
                end



            end
        end
    end

end

area

K;

%---

try chol(K);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end
sumKel=0; sumMel=0;

for i=1:nanchors
    for j=1:nanchors
        sumKel=sumKel+K(i,j);
        sumMel=sumMel+M(i,j);
    end
end     %end-of-IF detJ~= 0.
fprintf('SumKij=%12.5e  SumMij=%12.5e\n',sumKel,sumMel);

%     return

%     {
%% Stiffness matrix NGUYEN CONVENTIONAL FUNCTION
Kng=zeros(nanchors);  %initialize stiffness matrix
```

```
Mng=zeros(nanchors);  %initialize stiffness matrix

nel=noelmnts;   P(:,1)=XYctrl(1,:);   P(:,2)=XYctrl(2,:);   ncp=nanchors;
poly=p;
%    w=ones(nanchors,1);

IEN_array=IEN_calc(vertices,U_vec,V_vec,elmnts_pts,nanchors);


[Kng,areang]=FormK_H(IEN_array,P,Kng,nel,ncp,poly,weights',elmnts_pts,U
_vec,V_vec);
areang

try chol(Kng);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end

sumKel=0;
for i=1:nanchors
    for j=1:nanchors
        sumKel=sumKel+Kng(i,j);
    end
end     %end-of-IF detJ~= 0.
fprintf('SumKij=%12.5e \n',sumKel);


%  return

nanchorsEl=zeros(noelmnts,1);

nanchorsEl(:)=sum(IEN_array~=0,2);
p=3; q=3;

%% BÉZIER EXTRACTION

%---------- Bézier Operator Computation
C=zeros(max(nanchorsEl),(p+1)*(q+1),noelmnts);

for k=1:noelmnts
    nbas = nanchorsEl(k);
    C(1:nbas,:,k)=BE_Ce(p,q,nbas,IEN_array,k,elmnts_pts,U_vec,V_vec);
end
%----------


area2=0;
wb_e=zeros(noelmnts,(p+1)*(q+1));
```

```
K2=zeros(nanchors);
M2=zeros(nanchors);
Q=zeros((p+1)*(q+1),3,noelmnts); %Bézier control points

figure %Bézier control points and control mesh
axis equal
edof=0;
for k=1:noelmnts
    nbas = nanchorsEl(k);
    ianchory=zeros(1,nbas);
    %         nanchorsEl=size(IEN_array,2);
    %         Kel=0;  %initialize stiffness matrix
    %         Mel=0;  %initialize stiffness matrix
    ianchory(1:nbas)=nonzeros(IEN_array(k,:));

    XYctrloc=XYctrl(:,ianchory);



    W_e=diag(weights(:,ianchory));
    C_e=C(1:nbas,:,k);
    wb_e(k,:)=transpose(C_e)*weights(ianchory)';
    Wb_e=diag(wb_e(k,:));
    Q_e=inv(Wb_e)*transpose(C_e)*W_e*XYctrloc';
    edof=edof+size(Q_e,1);
    Q(:,:,k)=Q_e;
    igaussx=gi(1:ngaussx,ngaussx);
    igaussy=gi(1:ngaussy,ngaussy);

    Rbmat=zeros(nbas,length(igaussx),length(igaussy));
    dRmat_dx=zeros(nbas,length(igaussx),length(igaussy));
    dRmat_dy=zeros(nbas,length(igaussx),length(igaussy));

    shp=zeros(3,nbas);

    for igx=1:ngaussy

        for igy=1:ngaussx


[Rb,dR]=TSplineBasis(igaussx(igy),igaussy(igx),p,IEN_array,k,weights',C
_e);
            Rbmat(:,igx,igy)=Rb;
            dRmat_dx(:,igx,igy)=dR(1,:);
            dRmat_dy(:,igx,igy)=dR(2,:);
        end
    end

    for igx=1:ngaussy
        for igy=1:ngaussx
```

```
            %-----CONSTRUCT JACOBIAN AND ITS INVERSE
            %                 nel = nanchorsEl(k);    %number of control
points in the T-mesh
            shp(3,1:nbas)=Rbmat(1:nbas,igx,igy);
            shp(1,1:nbas)=dRmat_dx(1:nbas,igx,igy);
            shp(2,1:nbas)=dRmat_dy(1:nbas,igx,igy);
            for ii=1:2
                for jj=1:2
                    xs(ii,jj)=0;
                    for m=1:nbas
                        xs(ii,jj)=xs(ii,jj)+XYctrloc(ii,m)*shp(jj,m);
%Jacobian <------
                    end
                end
            end

            %---
            xsj=xs(1,1)*xs(2,2)-xs(1,2)*xs(2,1) ; %detJ
            sx(1,1)=xs(2,2)/xsj ; %Inverse of Jacobian
            sx(2,2)=xs(1,1)/xsj ;
            sx(1,2)=-xs(1,2)/xsj;
            sx(2,1)=-xs(2,1)/xsj;
            %-----FORM GLOBAL DERIVATIVES
            for ii=1:nbas
                tp       = shp(1,ii)*sx(1,1)+shp(2,ii)*sx(2,1);
                shp(2,ii)  =  shp(1,ii)*sx(1,2)+shp(2,ii)*sx(2,2);    %y-
derivative
                shp(1,ii)=tp ;                                        %x-
derivative
            end
            %---Store detJ:
            DETJacob(igx,igy)=xsj;

            %              DETJacob

            area2                =                area2                +
DETJacob(igx,igy)*ome(igy,ngaussx)*ome(igx,ngaussy);
            fprintf('Element=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

            for idof=1:nbas
                ii=IEN_array(k,idof);
                dNix=shp(1,idof);
                dNiy=shp(2,idof);
                for jdof=1:nbas
                    jj=IEN_array(k,jdof);
                    dNjx=shp(1,jdof);
                    dNjy=shp(2,jdof);

                    K2(ii,jj)=K2(ii,jj)+(dNix*dNjx+dNiy*dNjy)*...
```

```
DETJacob(igx,igy)*ome(igy,ngaussx)*ome(igx,ngaussy);

                    M2(ii,jj)=M2(ii,jj)+shp(3,idof)*shp(3,jdof)*...

DETJacob(igx,igy)*ome(igy,ngaussx)*ome(igx,ngaussy);
                end
            end
        end
    end
% end
%         if k==17
%             plot(Q(:,1,k),Q(:,2,k),'o', 'MarkerFaceColor', 'b');
%         else
scatter(Q(:,1,k),Q(:,2,k),10,'ko','filled');
%         end
hold on
plot(Q_e(1:p+1,1),Q_e(1:p+1,2),'-k')
plot(Q_e(1:p+1:end,1),Q_e(1:p+1:end,2),'-k')

plot(Q_e((q+1)*p+1:end,1),Q_e((q+1)*p+1:end,2),'-k')
plot(Q_e((p+1):p+1:end,1),Q_e((p+1):p+1:end,2),'-k')



end

    axis('equal')
%     text(xy_e(1,1,1),xy_e(1,1,2),int2str(nc));
% xlabel('x'); ylabel('y')
title('Bézier control points & control mesh')
% hold off

area2

figure %Bézier physical mesh
%           nc=0;

for e=1:noelmnts
    nbas = nanchorsEl(e);
    ianchory=zeros(1,nbas);
    ianchory(1:nbas)=nonzeros(IEN_array(e,:));
    XYctrloc=XYctrl(:,ianchory);

    W_e=diag(weights(:,ianchory));
    C_e=BE_Ce(p,q,nbas,IEN_array,e,elmnts_pts,U_vec,V_vec);
    wb_e(e,:)=transpose(C_e)*weights(ianchory)';
    Wb_e=diag(wb_e(e,:));
    Q_e=inv(Wb_e)*transpose(C_e)*W_e*XYctrloc';
    Q(:,:,e)=Q_e;
```

```
    nctrlX=5; nctrlY=5; nodes=nctrlX*nctrlY;

    dy=2/(nctrlY-1);
    dx=2/(nctrlX-1);
    xy_e=zeros(nctrlX,nctrlY,3);

    for j=1:nctrlY
        yij=(j-1)*dy-1;
        for i=1:nctrlX
            xij=(i-1)*dx-1;
            [Bbxi,dBxi]=BernsteinBasis(xij,p);
            [Bbeta,dBeta]=BernsteinBasis(yij,q);
            [Bb,dB]=BernsteinBasisBivariate(Bbxi,Bbeta,dBxi,dBeta,p);
            xy_e(i,j,:)=1/(wb_e(e,:)*Bb')*Q(:,:,e)'*Wb_e'*Bb';
            x_e(i,j)=xy_e(i,j,1);
            y_e(i,j)=xy_e(i,j,2);
        end
    end
    hold on
    %       plot(xy_e(:,:,1),xy_e(:,:,2), 'r.')
    plot(xy_e(1:nctrlX,1,1),xy_e(1:nctrlX,1,2),'-k')
    plot(xy_e(1:nctrlX,nctrlY,1),xy_e(1:nctrlX,nctrlY,2),'-k')

    plot(xy_e(1,1:end,1),xy_e(1,1:end,2),'-k')
    plot(xy_e(nctrlX,1:end,1),xy_e(nctrlX,1:end,2),'-k')
    axis('equal')

end

xlabel('x'); ylabel('y')
title('Bézier physical mesh')
hold off

try chol(K2);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end

sumKel=0;
for i=1:nanchors
    for j=1:nanchors
        sumKel=sumKel+K2(i,j);
    end
end     %end-of-IF detJ~= 0.
fprintf('SumKij=%12.5e \n',sumKel);

sumM2=0;
for i=1:nanchors
```

```
    for j=1:nanchors
        sumM2=sumM2+M2(i,j);
    end
end     %end-of-IF detJ~= 0.
fprintf('SumM2=%12.5e \n',sumM2);

%% IMPOSE BOUNDARY CONDITIONS
%{
   PARAMETRIC DOMAIN
%}
BC(1:57)=0; %initially all DOFs are unrestrained
BC(1:8)=1; BC(51:57)=1;
fi(1:8)=0; fi(51:57)=1000;

%---Arrange knowns and unknowns:
free_dofs = find(BC == 0);
fixed_dofs= find(BC == 1);
%
%
rhs = -K2(free_dofs,fixed_dofs)*fi(fixed_dofs)';
Coef(1:57)=0;
Coef(free_dofs) = K2(free_dofs,free_dofs) \ rhs;
%---Complete the BC in a single Coef:
Coef(fixed_dofs) = 0;
Coef(51:57)=fi(51:57);
%-----------------------------------------------------------------------
----

%% Error norm (L2): >>>
numerator=0;
denominator=0;
area=0;
Tm=1000;


%%    Error calculated on Gauss points of each element
ngaussx=2*p;
ngaussy=2*p;

for k=1:noelmnts
    igaussx=gi(1:ngaussx,ngaussx);
    igaussy=gi(1:ngaussy,ngaussy);

    nbas = nanchorsEl(k);
    ianchory=zeros(1,nbas);
    ianchory(1:nbas)=nonzeros(IEN_array(k,:));

    XYctrloc=XYctrl(:,ianchory);

    xy_e=zeros(ngaussy,ngaussx,3);
```

```
    W_e=diag(weights(:,ianchory));
    C_e=BE_Ce(p,q,nbas,IEN_array,k,elmnts_pts,U_vec,V_vec);
    wb_e(k,:)=transpose(C_e)*weights(ianchory)';
    Wb_e=diag(wb_e(k,:));
    Q_e=Wb_e\transpose(C_e)*W_e*XYctrloc';
    Q(:,:,k)=Q_e;

    Rbmat=zeros(nbas,ngaussx,ngaussy);

    for igx=1:ngaussy
        for igy=1:ngaussx


[Rb,dR]=TSplineBasis(igaussx(igy),igaussy(igx),p,IEN_array,k,weights',C
_e);
            Rbmat(:,igx,igy)=Rb;

            fprintf('Element=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

            [Bbxi,dBxi]=BernsteinBasis(igaussx(igy),p);
            [Bbeta,dBeta]=BernsteinBasis(igaussy(igx),q);
            [Bb,dB]=BernsteinBasisBivariate(Bbxi,Bbeta,dBxi,dBeta,p);

            xy_e(igx,igy,:)=1/(wb_e(k,:)*Bb')*Q(:,:,k)'*Wb_e'*Bb';
%physical space

            x_e(igx,igy)=xy_e(igx,igy,1);
            y_e(igx,igy)=xy_e(igx,igy,2);

            R=sqrt(x_e(igx,igy)^2+y_e(igx,igy)^2);
            Tex = (Tm/log(2))*log(R/1.5);

            T=0;
end
    end
    for igx=1:ngaussy
        for igy=1:ngaussx

            %-----CONSTRUCT JACOBIAN AND ITS INVERSE
            shp(3,1:nbas)=Rbmat(1:nbas,igx,igy);
            shp(1,1:nbas)=dRmat_dx(1:nbas,igx,igy);
            shp(2,1:nbas)=dRmat_dy(1:nbas,igx,igy);
            for ii=1:2
                for jj=1:2
                    xs(ii,jj)=0;
                    for m=1:nbas
                        xs(ii,jj)=xs(ii,jj)+XYctrloc(ii,m)*shp(jj,m);
%Jacobian <------
                    end
                end
            end

            %---
```

```
            xsj=xs(1,1)*xs(2,2)-xs(1,2)*xs(2,1) ; %detJ
            sx(1,1)=xs(2,2)/xsj ; %Inverse of Jacobian
            sx(2,2)=xs(1,1)/xsj ;
            sx(1,2)=-xs(1,2)/xsj;
            sx(2,1)=-xs(2,1)/xsj;
            %-----FORM GLOBAL DERIVATIVES
            for ii=1:nbas
                tp       = shp(1,ii)*sx(1,1)+shp(2,ii)*sx(2,1);
                shp(2,ii) = shp(1,ii)*sx(1,2)+shp(2,ii)*sx(2,2);   %y-
derivative
                shp(1,ii)=tp ;                                     %x-
derivative
            end
            %---Store detJ:
            DETJacob(igx,igy)=xsj;
        end
    end

            for s=1:length(ianchory)
                T = T +  Rbmat(s,igx,igy)*Coef(ianchory(s));
            end
            numerator=numerator+(T-
Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);

denominator=denominator+(Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);
        end
    end

end
fprintf('ERROR                          NORM                          (in
percent)=%10.4f\n',100*sqrt(numerator/denominator));%error-norm     [in
(%)]

%% DETERMINE THE TEMPERATURE AT % UNIFORM POSITIONS PER EACH ELEMENT'S
DIRECTION:
nctrlX=5; nctrlY=5; nodes=nctrlX*nctrlY;

nc=0;
Temp(1:nodes*noelmnts)=0;
figure %Solution on the physical domain
shading interp
for e=1:noelmnts
    nbas = nanchorsEl(e);
    ianchory=zeros(1,nbas);
    ianchory(1:nbas)=nonzeros(IEN_array(e,:));
    XYctrloc=XYctrl(:,ianchory);

    W_e=diag(weights(:,ianchory));
    C_e=BE_Ce(p,q,nbas,IEN_array,e,elmnts_pts,U_vec,V_vec);
    wb_e(e,:)=transpose(C_e)*weights(ianchory)';
    Wb_e=diag(wb_e(e,:));
    Q_e=inv(Wb_e)*transpose(C_e)*W_e*XYctrloc';
    Q(:,:,e)=Q_e;
```

```
    X=zeros(nctrlX,nctrlY);  Y=zeros(nctrlX,nctrlY);

    xy_e=zeros(nctrlX,nctrlY,3);
    x_e=zeros(nctrlX,nctrlY);
    y_e=zeros(nctrlX,nctrlY);
    SOL=zeros(nctrlX,nctrlY);

    dy=2/(nctrlY-1);
    dx=2/(nctrlX-1);



    Rbmat=zeros(nbas,ngaussx,ngaussy);
    for j=1:nctrlY
        yij=(j-1)*dy-1;
        for i=1:nctrlX
            xij=(i-1)*dx-1;
            nc=nc+1;
            [Bbxi,dBxi]=BernsteinBasis(xij,p);
            [Bbeta,dBeta]=BernsteinBasis(yij,q);
            [Bb,dB]=BernsteinBasisBivariate(Bbxi,Bbeta,dBxi,dBeta,p);

            xy_e(i,j,:)=1/(wb_e(e,:)*Bb')*Q(:,:,e)'*Wb_e'*Bb';
            x_e(i,j)=xy_e(i,j,1);
            y_e(i,j)=xy_e(i,j,2);

            [Rb,dR]=TSplineBasis(xij,yij,p,IEN_array,e,weights',C_e);
            %                       Rbmat(:,igx,igy)=Rb;

            Temp(nc)=Coef(ianchory)*Rb';
            %                       T = T +  Nip(i,igx,igy)*Coef(i);

            for k=1:length(ianchory)
                SOL(i,j)=SOL(i,j) + Rb(k) * Coef(ianchory(k));

            end
            Tex(i,j)=(Tm/log(2))*log(sqrt(x_e(i,j)^2+y_e(i,j)^2)/1.5);
            Texact(nc)=Tex(i,j);
        end
    end

    %       plot(xy_e(:,:,1),xy_e(:,:,2), 'r.')
    %       hold off
    %       figure
    hold on
    surf(x_e,y_e,SOL)
end
xlabel('x'); ylabel('y'); zlabel('z')
shading interp
```

```
title({'Solution on physical domain'}) %,[' Partition points in each
element: ', num2str(nctrlX),' in x dir &', num2str(nctrlY), ' in y dir']})
hold off
%  return
Texact;
fprintf('                    \n');
fprintf('                    \n');
fprintf('********   R E S U L T S   *******\n');
fprintf('Node     Temperature      Texact\n');
for i=1:nc
    fprintf('%3i    %10.4f      %10.4f\n',i,Temp(i),Texact(i));
end
```

A.3 Ορθογώνιο χωρίο

A.3.1 Περιγραφή

Στα παρακάτω προγράμματα ορίζεται γεωμετρικά το ορθογώνιο χωρίο διάστασης $a \times b$ με χρήση των συναρτήσεων Bézier, B-spline και T-spline, αντίστοιχα, ενώ επιλύεται το πρόβλημα Laplace, το οποίο ορίστηκε στο Κεφάλαιο 0, με τις αντίστοιχες συναρτήσεις βάσης.

A.3.2 Κύριος Κώδικας

A.3.2.1 Επίλυση με συναρτήσεις Bernstein (DoF=16)

```
      %Test-T-Spline
%---
clear all
clc;
t = tSpline([],3,[],[],[],[],[]);

%{
13oo14----------------15oo16
9oo10-----------------11oo12
  ||                    ||
  ||                    ||
  ||                    ||
  ||                    ||
  ||                    ||
  ||                    ||
 5oo6------------------7oo8
  oo---------------------oo
  12                    34

%}
a=3; b=12;
vertices = [ kVertex(0,0,0)        %anchor 1
             kVertex(0,0,0)        %anchor 2
             kVertex(a,0,0)        %anchor 3
             kVertex(a,0,0)        %anchor 4
```

```
              kVertex(0,0,0)        %anchor 5
              kVertex(0,0,0)        %anchor 6
              kVertex(a,0,0)        %anchor 7
              kVertex(a,0,0)        %anchor 8
              kVertex(0,b,0)       %anchor 9
              kVertex(0,b,0)       %anchor 10
              kVertex(a,b,0)       %anchor 11
              kVertex(a,b,0)       %anchor 12
              kVertex(0,b,0)       %anchor 13
              kVertex(0,b,0)       %anchor 14
              kVertex(a,b,0)       %anchor 15
              kVertex(a,b,0)];     %anchor 16


%% Edges-------------------

vertices(1).cnctRight(vertices(2));
vertices(1).cnctTop(vertices(5));

vertices(2).cnctTop(vertices(6));
vertices(2).cnctRight(vertices(3));
vertices(2).cnctLeft(vertices(1));

vertices(3).cnctTop(vertices(7));
vertices(3).cnctRight(vertices(4));
vertices(3).cnctLeft(vertices(2));

vertices(4).cnctTop(vertices(8));
vertices(4).cnctLeft(vertices(3));



vertices(5).cnctTop(vertices(9));
vertices(5).cnctRight(vertices(6));
vertices(5).cnctBottom(vertices(1));

vertices(6).cnctTop(vertices(10));
vertices(6).cnctRight(vertices(7));
vertices(6).cnctBottom(vertices(2));
vertices(6).cnctLeft(vertices(5));

vertices(7).cnctTop(vertices(11));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctBottom(vertices(3));
vertices(7).cnctLeft(vertices(6));

vertices(8).cnctTop(vertices(12));
vertices(8).cnctBottom(vertices(4));
vertices(8).cnctLeft(vertices(7));

vertices(9).cnctTop(vertices(13));
```

```
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(5));

vertices(10).cnctTop(vertices(14));
vertices(10).cnctRight(vertices(11));
vertices(10).cnctBottom(vertices(6));
vertices(10).cnctLeft(vertices(9));

vertices(11).cnctTop(vertices(15));
vertices(11).cnctRight(vertices(12));
vertices(11).cnctBottom(vertices(7));
vertices(11).cnctLeft(vertices(10));

vertices(12).cnctTop(vertices(16));
vertices(12).cnctBottom(vertices(8));
vertices(12).cnctLeft(vertices(11));

vertices(13).cnctRight(vertices(14));
vertices(13).cnctBottom(vertices(9));

vertices(14).cnctRight(vertices(15));
vertices(14).cnctBottom(vertices(10));
vertices(14).cnctLeft(vertices(13));

vertices(15).cnctRight(vertices(16));
vertices(15).cnctBottom(vertices(11));
vertices(15).cnctLeft(vertices(14));

vertices(16).cnctBottom(vertices(12));
vertices(16).cnctLeft(vertices(15));

t.kVertices = vertices;

%t.printPreImagePaper();
t.updateKnotVecs();

sampleCount = 20;
ps = linspace(0,1,sampleCount);
pt = linspace(0,1,sampleCount);



p=3;
nanchors=16;

%% Determine the step we move horizontally and vertically from anchor:
    nc=0;
    if(mod(p,2)==0)
        step=p/2+1;
        ncolumns=2*step;
```

```
    else
        step=(p+1)/2;
        ncolumns=2*step+1;
    end

numberOfVertices = size(vertices,1);

for i=1:numberOfVertices
   U_vec(i,1:ncolumns)=vertices(i,1).knotVec(1,:);  %ncolumns=5, for p =
3.
   V_vec(i,1:ncolumns)=vertices(i,1).knotVec(2,:);  %ncolumns=5, for p =
3.
end
%%

%% DETERMINE CONTROL POINTS SO AS TO PRESERVE LINEAR MAPPING: RIPFUNNORM
%     x = [vertices.s];   %x-coordinates of vertices
%     y = [vertices.t];   %y-coordinates of vertices
%     xyzctrl=[vertices(:).cPoint];
%         xctrl=xyzctrl(1,:);  yctrl=xyzctrl(2,:);  zctrl=xyzctrl(3,:);
%POLYTIMI
%---
% % %     figure
% % %     plot(x,y,'bo-')
%Define 38(or ??:least-squares-solution) arbitrary points and then fit
the control points:
% x(ξ,η)=SUM [Nip(ξ,η).Xcontrols (1)
% y(ξ,η)=SUM [Nip(ξ,η).Ycontrols (2)
%---
    nx=7; ny=7; %normally: nx=5; ny=6;  5   5
    dx=a/nx; dy=b/ny;
    nc=0;
    for i=1:ny+1
        ht=(i-1)*dy;
        for j=1:nx+1
            nc=nc+1;
            cs=(j-1)*dx;
                     [~,~,Nip,dNip_dx,dNip_dy] = t.blendfunNorm(cs,ht);

            Amat(nc,1:nanchors)=Nip;
            RhsX(nc)=cs; %rhs in eq.(1)
            RhsY(nc)=ht; %rhs in eq.(2)
        end
    end
%     Amat
% RhsX
% RhsY
    Xctrl = Amat \ RhsX';   %eq.(1)
    Yctrl = Amat \ RhsY';   %eq.(2)
% figure
% hold on
```

```
% plot(xctrl,yctrl,'m+')
hold on
plot(Xctrl,Yctrl,'rx')
% legend('Polytimi','CP','Location','best')
axis equal

    XYctrl(1,1:nanchors)=Xctrl(1:nanchors);
    XYctrl(2,1:nanchors)=Yctrl(1:nanchors);
    XYctrl(3,1:nanchors)=zeros(1,nanchors);

    for i=1:length(vertices)
        vertices(i).cPoint=XYctrl(1:3,i);
    end
    t.kVertices=vertices;
    t.updateKnotVecs;
%---- Ctrl Points

C=zeros(length(vertices),3);
for i=1:length(vertices)
%         Xctrl(i)=vertices(i).cPoint(1);
%         Yctrl(i)=vertices(i).cPoint(2);
        C(i,:)=t.kVertices(i).cPoint(1:3);
%         ST(i,:)=[t.kVertices(i)];



end
plot(C(:,1),C(:,2),'ro');
nctrls = size(C(:,1),1);
dsx=a/50; dsy=b/50;
for i=1:nctrls
    text(C(i,1)+dsx,C(i,2)+dsy,int2str(i),'Color','b');
end
% title('CONTROL POINTS')
axis equal
grid on

%%%%STIFFNESS MATRIX (LAUBE FUNS)

%%Calculate the patches:
numberOfPatches=21;
patches=t.patches_calc(numberOfPatches);

% Gauss points and weights:
    [gi,ome] = pgau_rectangle_FULL;
    ngaussx=p+1;   %ceil((1+p^2)/2);
    ngaussy=ngaussx;
 %% Stiffness matrix
    K=zeros(nanchors);  %initialize stiffness matrix
    M=zeros(nanchors);  %initialize stiffness matrix
    area=0;
```

```
% % %
    for k=1:numberOfPatches
        %Corner points:
        xA=patches(k,1); yA=patches(k,2);
        xB=patches(k,3); yB=patches(k,4);
        xC=patches(k,5); yC=patches(k,6);
        xD=patches(k,7); yD=patches(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
        taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
        taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);

B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y]                          =
t.blendfun(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:);
dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end

 fprintf('sumB_mat=%3i\n',sum(B_mat(1,2,:)));

        for igx=1:ngaussy

            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
        fprintf('Patch=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

        for i=1:nanchors
                    dNix=dB_xmat(igx,igy,i);
                    dNiy=dB_ymat(igx,igy,i);
                    for j=1:nanchors
                        dNjx=dB_xmat(igx,igy,j);
                        dNjy=dB_ymat(igx,igy,j);
                        K(i,j)=K(i,j)+(dNix*dNjx+dNiy*dNjy)*...

detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                        M(i,j)=M(i,j)+B_mat(igx,igy,i)                *
B_mat(igx,igy,j)*...
```

```
detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                    end
         end


            end
        end
    end
    end
    area

    K;
%---

try chol(M);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end
sumM=0; sumK=0;

    for i=1:nanchors
        for j=1:nanchors
            sumK=sumK+K(i,j);
            sumM=sumM+M(i,j);
        end
    end     %end-of-IF detJ~= 0.
    fprintf('SumKij=%12.5e  SumMij=%12.5e\n',sumK,sumM);

%     return
%}

%% IMPOSE BOUNDARY CONDITIONS
%{
 D                    C
13oo14----------------15oo16
9oo10-----------------11oo12
  ||                    ||
  ||                    ||
  ||                    ||
  ||                    ||
  ||                    ||
  ||                    ||
 5oo6------------------7oo8
  oo---------------------oo
  12                    34
   A                     B
%}
    BC(1:16)=0; %initially all DOFs are unrestrained
```

```
   BC(1:4)=1; BC(8:4:12)=1;   BC(13:16)=1;
    fi(1:4)=0; fi(8:4:12)=0;  fi(13:16)=0;
%---Find the coefficients along the side DC using equally-spaced spans:
    knots_DC = [0,0,0,0,3,3,3,3];
    xtest_DC=linspace(0,3,4);
    Utest_DC=1000*cos(pi*xtest_DC/6);
    nstep=1; %We need only two terms(Nip and first derivative).
    p=3;
    colmatTEST_DC=spcol(knots_DC,p+1,brk2knt(xtest_DC,nstep));  %1  term
(Nip-only)
    i1y=size(colmatTEST_DC,1);
    BasisTEST_DC = colmatTEST_DC(1:nstep:i1y,:);
%---Using the above matrix, determine the control points:
    XcontrolP_DC = BasisTEST_DC \ xtest_DC';
%---Using the above matrix, determine the coefficients:
    Coef_DC = BasisTEST_DC \ Utest_DC';
    fi(13:16)=Coef_DC;
%---Quality of representing the BC along DC:
    nseg=100;
    tauseg=linspace(0,3,nseg+1);
    colmatseg=spcol(knots_DC,p+1,brk2knt(tauseg,nstep));  %1  term  (Nip-
only)
    i1y=size(colmatseg,1);
    Basisseg = colmatseg(1:nstep:i1y,:);
    Useg=Basisseg*Coef_DC;
%
    figure
    plot(tauseg,Useg,'bo')
    hold on
    plot(tauseg,1000*cos(pi*tauseg/6),'r','Linewidth',2)
    xlabel('X')
    ylabel('T(x)')
    title('CALCULATED versus EXACT along DC')
    legend('Approximation','Exact','Location','best')
%---Arrange knowns and unknowns:
    free_dofs = find(BC == 0);
    fixed_dofs= find(BC == 1);
%



    rhs = -K(free_dofs,fixed_dofs)*fi(fixed_dofs)';
    Coef(1:16)=0;
    Coef(free_dofs) = K(free_dofs,free_dofs) \ rhs;
%---Complete the BC in a single Coef:
    Coef(fixed_dofs) = 0;
    Coef(13:16)=Coef_DC;
%-----------------------------------------------------------------------
----

%% Error norm (L2): >>>
    numerator=0;
```

```
    denominator=0;
    area=0;
    Tm=1000; a=3; b=12;
    %% NEW-STYLE (Provatidis: Friday morning 6:47, 7/2/2020):
    ngaussx=p+1;
    ngaussy=p+1;
    for k=1:numberOfPatches
        %Corner points:
        xA=patches(k,1); yA=patches(k,2);
        xB=patches(k,3); yB=patches(k,4);
        xC=patches(k,5); yC=patches(k,6);
        xD=patches(k,7); yD=patches(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
            taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
            taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);

B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y]                          =
t.blendfunNorm(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:);  %matrix  4x4x16  (4x4  gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end
        %
        for igx=1:ngaussy
            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                xgpt = taugaussx(igy); ygpt = taugaussy(igx);
                Tex          =          Tm*sinh(pi*ygpt/2/a)          /
sinh(pi*b/2/a)*cos(pi*xgpt/2/a);
               T=0;
               for i=1:nanchors
                  T = T +  B_mat(igx,igy,i)*Coef(i);
               end
           numerator=numerator+(T-
Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);
```

```
denominator=denominator+(Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);
            end
        end
        end
    end
    fprintf('ERROR                        NORM                        (in
percent)=%10.4f\n',100*sqrt(numerator/denominator));%error-norm      [in
(%)]
%---

%% DETERMINE THE TEMPERATURE AT SIX UNIFORM POSITIONS PER DIRECTION:
    nctrlX=4; nctrlY=4; nodes=nctrlX*nctrlY;
%---Approximate solution:
    Temp(1:nodes)=0;

    dy=b/(nctrlY-1); dht=1/(nctrlY-1);
    dx=a/(nctrlX-1); dcs=1/(nctrlX-1);
    nc=0;
    for j=1:nctrlY
        yij=(j-1)*dy; ht=(j-1)*dht;
        for i=1:nctrlX
            xij=(i-1)*dx; cs=(i-1)*dcs;
            nc=nc+1;
             Texact(nc)=Tm*sinh(pi*yij/2/a)                            /
sinh(pi*b/2/a)*cos(pi*xij/2/a);


                    [~, ~,Nip,~,~] = t.blendfunNorm(xij,yij);
                    Temp(nc)=Coef*Nip;




        end
    end
%
    fprintf('                    \n');
    fprintf('                    \n');
    fprintf('********   R E S U L T S   *******\n');
    fprintf('Node     Temperature      Texact\n');
    for i=1:nodes
        fprintf('%3i    %10.4f      %10.4f\n',i,Temp(i),Texact(i));
    end
```

A.3.2.2 Επίλυση με συναρτήσεις B-spline με έναν εσωτερικό κόμβο ανά κατεύθυνση (DoF=25)

```
%Test-T-Spline
%---
```

```
clear all
clc;
t = tSpline([],3,[],[],[],[],[]);

%{
21oo22-------23o-----------24oo25
16oo17-------18o----------19oo20
  ||           |            ||
  ||           |            ||
16oo17-------19o---------20oo21
  ||           |            ||
  ||           |            ||
11oo12-------13o----------14oo15
  ||           |            ||
  ||           |            ||
  ||           |            ||
  ||           |            ||
 6oo7---------8o-----------9oo10
  oo-----------o------------oo
  12           3        45

%}
a=3; b=12;
vertices = [ kVertex(0,0,0)       %anchor 1
             kVertex(0,0,0)       %anchor 2
             kVertex(a/2,0,0)     %anchor 3
             kVertex(a,0,0)       %anchor 4
             kVertex(a,0,0)       %anchor 5
             kVertex(0,0,0)       %anchor 6
             kVertex(0,0,0)       %anchor 7
             kVertex(a/2,0,0)     %anchor 8
             kVertex(a,0,0)       %anchor 9
             kVertex(a,0,0)       %anchor 10
             kVertex(0,b/2,0)       %anchor 11
             kVertex(0,b/2,0)       %anchor 12
             kVertex(a/2,b/2,0)     %anchor 13
             kVertex(a,b/2,0)       %anchor 14
             kVertex(a,b/2,0)       %anchor 15
             kVertex(0,b,0)      %anchor 16
             kVertex(0,b,0)      %anchor 17
             kVertex(a/2,b,0)    %anchor 18
             kVertex(a,b,0)      %anchor 19
             kVertex(a,b,0)      %anchor 20
             kVertex(0,b,0)      %anchor 21
             kVertex(0,b,0)      %anchor 22
             kVertex(a/2,b,0)    %anchor 23
             kVertex(a,b,0)      %anchor 24
             kVertex(a,b,0)];    %anchor 25


%% Edges-------------------
```

```
vertices(1).cnctRight(vertices(2));
vertices(1).cnctTop(vertices(6));

vertices(2).cnctTop(vertices(7));
vertices(2).cnctRight(vertices(3));
vertices(2).cnctLeft(vertices(1));

vertices(3).cnctTop(vertices(8));
vertices(3).cnctRight(vertices(4));
vertices(3).cnctLeft(vertices(2));

vertices(4).cnctTop(vertices(9));
vertices(4).cnctRight(vertices(5));
vertices(4).cnctLeft(vertices(3));

vertices(5).cnctTop(vertices(10));
vertices(5).cnctLeft(vertices(4));

vertices(6).cnctTop(vertices(11));
vertices(6).cnctRight(vertices(7));
vertices(6).cnctBottom(vertices(1));

vertices(7).cnctTop(vertices(12));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctBottom(vertices(2));
vertices(7).cnctLeft(vertices(6));

vertices(8).cnctTop(vertices(13));
vertices(8).cnctRight(vertices(9));
vertices(8).cnctBottom(vertices(3));
vertices(8).cnctLeft(vertices(7));

vertices(9).cnctTop(vertices(14));
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(4));
vertices(9).cnctLeft(vertices(8));

vertices(10).cnctTop(vertices(15));
vertices(10).cnctBottom(vertices(5));
vertices(10).cnctLeft(vertices(9));

vertices(11).cnctTop(vertices(16));
vertices(11).cnctRight(vertices(12));
vertices(11).cnctBottom(vertices(6));

vertices(12).cnctTop(vertices(17));
vertices(12).cnctRight(vertices(13));
vertices(12).cnctBottom(vertices(7));
vertices(12).cnctLeft(vertices(11));
```

```
vertices(13).cnctTop(vertices(18));
vertices(13).cnctRight(vertices(14));
vertices(13).cnctBottom(vertices(8));
vertices(13).cnctLeft(vertices(12));


vertices(14).cnctTop(vertices(19));
vertices(14).cnctRight(vertices(15));
vertices(14).cnctBottom(vertices(9));
vertices(14).cnctLeft(vertices(13));

vertices(15).cnctTop(vertices(20));
vertices(15).cnctBottom(vertices(10));
vertices(15).cnctLeft(vertices(14));

vertices(16).cnctTop(vertices(21));
vertices(16).cnctRight(vertices(17));
vertices(16).cnctBottom(vertices(11));

vertices(17).cnctTop(vertices(22));
vertices(17).cnctRight(vertices(18));
vertices(17).cnctBottom(vertices(12));
vertices(17).cnctLeft(vertices(16));

vertices(18).cnctTop(vertices(23));
vertices(18).cnctRight(vertices(19));
vertices(18).cnctBottom(vertices(13));
vertices(18).cnctLeft(vertices(17));

vertices(19).cnctTop(vertices(24));
vertices(19).cnctRight(vertices(20));
vertices(19).cnctBottom(vertices(14));
vertices(19).cnctLeft(vertices(18));

vertices(20).cnctTop(vertices(25));
vertices(20).cnctBottom(vertices(15));
vertices(20).cnctLeft(vertices(19));

vertices(21).cnctRight(vertices(22));
vertices(21).cnctBottom(vertices(16));

vertices(22).cnctRight(vertices(23));
vertices(22).cnctBottom(vertices(17));
vertices(22).cnctLeft(vertices(21));

vertices(23).cnctRight(vertices(24));
vertices(23).cnctBottom(vertices(18));
vertices(23).cnctLeft(vertices(22));
```

```
vertices(24).cnctRight(vertices(25));
vertices(24).cnctBottom(vertices(19));
vertices(24).cnctLeft(vertices(23));

vertices(25).cnctBottom(vertices(20));
vertices(25).cnctLeft(vertices(24));


t.kVertices = vertices;

%% Control Points
%{
vertices(1).cPoint = [0;0;0];
vertices(2).cPoint = [a/6;0;0];
vertices(3).cPoint = [3*a/6;0;0];
vertices(4).cPoint = [5*a/6;0;0];
vertices(5).cPoint = [a;0;0];

vertices(6).cPoint = [0;b/6;0];
vertices(7).cPoint = [a/6;b/6;0];
vertices(8).cPoint = [3*a/6;b/6;0];
vertices(9).cPoint = [5*a/6;b/6;0];
vertices(10).cPoint = [a;b/6;0];

vertices(11).cPoint = [0;3*b/6;0];
vertices(12).cPoint = [a/6;3*b/6;0];
vertices(13).cPoint = [3*a/6;3*b/6;0];
vertices(14).cPoint = [5*a/6;3*b/6;0];
vertices(15).cPoint = [a;3*b/6;0];

vertices(16).cPoint = [0;5*b/6;0];
vertices(17).cPoint = [a/6;5*b/6;0];
vertices(18).cPoint = [3*a/6;5*b/6;0];
vertices(19).cPoint = [5*a/6;5*b/6;0];
vertices(20).cPoint = [a;5*b/6;0];

vertices(21).cPoint = [0;b;0];
vertices(22).cPoint = [a/6;b;0];
vertices(23).cPoint = [3*a/6;b;0];
vertices(24).cPoint = [5*a/6;b;0];
vertices(25).cPoint = [a;b;0];

%}

%t.printPreImagePaper();
t.updateKnotVecs();

sampleCount = 20;
ps = linspace(0,1,sampleCount);
```

```
pt = linspace(0,1,sampleCount);



p=3;
nanchors=25;

%% Determine the step we move horizontally and vertically from anchor:
    nc=0;
    if(mod(p,2)==0)
        step=p/2+1;
        ncolumns=2*step;
    else
        step=(p+1)/2;
        ncolumns=2*step+1;
    end

numberOfVertices = size(vertices,1);

for i=1:numberOfVertices
   U_vec(i,1:ncolumns)=vertices(i,1).knotVec(1,:);  %ncolumns=5, for p =
3.
   V_vec(i,1:ncolumns)=vertices(i,1).knotVec(2,:);  %ncolumns=5, for p =
3.
end
%%

%% DETERMINE CONTROL POINTS SO AS TO PRESERVE LINEAR MAPPING: RIPFUNNORM
%     x = [vertices.s];   %x-coordinates of vertices
%     y = [vertices.t];   %y-coordinates of vertices
%     xyzctrl=[vertices(:).cPoint];
%         xctrl=xyzctrl(1,:);  yctrl=xyzctrl(2,:);  zctrl=xyzctrl(3,:);
%POLYTIMI
%---
% % %     figure
% % %     plot(x,y,'bo-')
%Define 38(or ??:least-squares-solution) arbitrary points and then fit
the control points:
% x(ξ,η)=SUM [Nip(ξ,η).Xcontrols (1)
% y(ξ,η)=SUM [Nip(ξ,η).Ycontrols (2)
%---
    nx=7; ny=7; %normally: nx=5; ny=6;  5   5
    dx=a/nx; dy=b/ny;
    nc=0;
    for i=1:ny+1
        ht=(i-1)*dy;
        for j=1:nx+1
            nc=nc+1;
            cs=(j-1)*dx;
     [~,~,Nip,dNip_dx,dNip_dy]=t.blendfunNorm(cs,ht);
            Amat(nc,1:nanchors)=Nip;
```

```
            RhsX(nc)=cs; %rhs in eq.(1)
            RhsY(nc)=ht; %rhs in eq.(2)
        end
    end
%     Amat
% RhsX
% RhsY
    Xctrl = Amat \ RhsX';   %eq.(1)
    Yctrl = Amat \ RhsY';   %eq.(2)
% figure
% hold on
% plot(xctrl,yctrl,'m+')
hold on
plot(Xctrl,Yctrl,'rx')
% legend('Polytimi','CP','Location','best')
axis equal

    XYctrl(1,1:nanchors)=Xctrl(1:nanchors);
    XYctrl(2,1:nanchors)=Yctrl(1:nanchors);
    XYctrl(3,1:nanchors)=zeros(1,nanchors);

    for i=1:length(vertices)
        vertices(i).cPoint=XYctrl(1:3,i);
    end
    t.kVertices=vertices;
    t.updateKnotVecs;
%---- Ctrl Points

C=zeros(length(vertices),3);
for i=1:length(vertices)
%         Xctrl(i)=vertices(i).cPoint(1);
%         Yctrl(i)=vertices(i).cPoint(2);
        C(i,:)=t.kVertices(i).cPoint(1:3);
%         ST(i,:)=[t.kVertices(i)];


end
plot(C(:,1),C(:,2),'ro');
nctrls = size(C(:,1),1);
dsx=a/50; dsy=b/50;
for i=1:nctrls
    text(C(i,1)+dsx,C(i,2)+dsy,int2str(i),'Color','b');
end
% title('CONTROL POINTS')
axis equal
grid on
%%%%STIFFNESS MATRIX (LAUBE FUNS)

%%Calculate the patches:
numberOfPatches=21;
patches=t.patches_calc(numberOfPatches);
```

```
% Gauss points and weights:
    [gi,ome] = pgau_rectangle_FULL;
    ngaussx=p+1;   %ceil((1+p^2)/2);
    ngaussy=ngaussx;
 %% Stiffness matrix
    K=zeros(nanchors);  %initialize stiffness matrix
    M=zeros(nanchors);  %initialize stiffness matrix
    area=0;
% % %
% % %         taugaussx=(1+gi(1:ngaussx,ngaussx))/2;  %0<Ugauss<1
% % %         dRipx=
% % %         taugaussy=(1+gi(1:ngaussy,ngaussy))/2;  %0<Vgauss<1
% % %
    for k=1:numberOfPatches
        %Corner points:
        xA=patches(k,1); yA=patches(k,2);
        xB=patches(k,3); yB=patches(k,4);
        xC=patches(k,5); yC=patches(k,6);
        xD=patches(k,7); yD=patches(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
        taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
        taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);
%                                           [Nip,dNip_dx,dNip_dy]     =
Ripfun(nanchors,U_vec,V_vec,ncolumns,p,...
%                                        taugaussx,taugaussy);


B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y]                          =
t.blendfun(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:);  %matrix  4x4x16  (4x4  gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);


            end
        end

 fprintf('sumB_mat=%3i\n',sum(B_mat(1,2,:)));
```

```
        for igx=1:ngaussy

            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
        fprintf('Patch=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

        for i=1:nanchors
                    dNix=dB_xmat(igx,igy,i);
                    dNiy=dB_ymat(igx,igy,i);
                    for j=1:nanchors
                        dNjx=dB_xmat(igx,igy,j);
                        dNjy=dB_ymat(igx,igy,j);
                        K(i,j)=K(i,j)+(dNix*dNjx+dNiy*dNjy)*...

detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                        M(i,j)=M(i,j)+B_mat(igx,igy,i)                 *
B_mat(igx,igy,j)*...

detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                    end
         end



            end
        end
    end
    end
    area

    K;
%---

try chol(M);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end
sumM=0; sumK=0;

    for i=1:nanchors
        for j=1:nanchors
            sumK=sumK+K(i,j);
            sumM=sumM+M(i,j);
        end
    end     %end-of-IF detJ~= 0.
    fprintf('SumKij=%12.5e  SumMij=%12.5e\n',sumK,sumM);

%     return
%}
```

```
%% IMPOSE BOUNDARY CONDITIONS
%{
 D                     C
21oo22----23o------24oo25
16oo17----18o------19oo20
  ||        |        ||
  ||        |        ||
  ||        |        ||
  ||        |        ||
11oo12----13o------14oo15
  ||        |        ||
  ||        |        ||
  ||        |        ||
  ||        |        ||
 6oo7------8o-------9oo10
  oo--------o--------oo
A 12        3        45  B
%}
    BC(1:25)=0; %initially all DOFs are unrestrained
    BC(1:5)=1; BC(10:5:20)=1;   BC(21:25)=1;
    fi(1:5)=0; fi(10:5:20)=0;  fi(21:25)=0;
%---Find the coefficients along the side DC using equally-spaced spans:
    knots_DC = [0,0,0,0,1.5,3,3,3,3];
    xtest_DC=linspace(0,3,5);
    Utest_DC=1000*cos(pi*xtest_DC/6);
    nstep=1; %We need only two terms(Nip and first derivative).
    p=3;
    colmatTEST_DC=spcol(knots_DC,p+1,brk2knt(xtest_DC,nstep));  %1  term
(Nip-only)
    i1y=size(colmatTEST_DC,1);
    BasisTEST_DC = colmatTEST_DC(1:nstep:i1y,:);
%---Using the above matrix, determine the control points:
    XcontrolP_DC = BasisTEST_DC \ xtest_DC';
%---Using the above matrix, determine the coefficients:
    Coef_DC = BasisTEST_DC \ Utest_DC';
    fi(21:25)=Coef_DC;
%---Quality of representing the BC along DC:
    nseg=100;
    tauseg=linspace(0,3,nseg+1);
    colmatseg=spcol(knots_DC,p+1,brk2knt(tauseg,nstep));  %1  term  (Nip-
only)
    i1y=size(colmatseg,1);
    Basisseg = colmatseg(1:nstep:i1y,:);
    Useg=Basisseg*Coef_DC;
%
    figure
    plot(tauseg,Useg,'bo')
    hold on
    plot(tauseg,1000*cos(pi*tauseg/6),'r','Linewidth',2)
    xlabel('X')
```

```
    ylabel('T(x)')
    title('CALCULATED versus EXACT along DC')
    legend('Approximation','Exact','Location','best')
%---Arrange knowns and unknowns:
    free_dofs = find(BC == 0);
    fixed_dofs= find(BC == 1);
%
%
    rhs = -K(free_dofs,fixed_dofs)*fi(fixed_dofs)';
    Coef(1:25)=0;
    Coef(free_dofs) = K(free_dofs,free_dofs) \ rhs;
%---Complete the BC in a single Coef:
    Coef(fixed_dofs) = 0;
    Coef(21:25)=Coef_DC;
%-----------------------------------------------------------------------
----

%% Error norm (L2): >>>
    numerator=0;
    denominator=0;
    area=0;
    Tm=1000; a=3; b=12;
    %% NEW-STYLE (Provatidis: Friday morning 6:47, 7/2/2020):
    ngaussx=p+1;
    ngaussy=p+1;
    for k=1:numberOfPatches
        %Corner points:
        xA=patches(k,1); yA=patches(k,2);
        xB=patches(k,3); yB=patches(k,4);
        xC=patches(k,5); yC=patches(k,6);
        xD=patches(k,7); yD=patches(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
            taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
            taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);

B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y]                          =
t.blendfunNorm(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:);  %matrix  4x4x16  (4x4  gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
```

```
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end
        %
        for igx=1:ngaussy
            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                xgpt = taugaussx(igy); ygpt = taugaussy(igx);
                Tex          =          Tm*sinh(pi*ygpt/2/a)          /
sinh(pi*b/2/a)*cos(pi*xgpt/2/a);
               T=0;
               for i=1:nanchors
                  T = T +  B_mat(igx,igy,i)*Coef(i);
               end
           numerator=numerator+(T-
Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);

denominator=denominator+(Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);
            end
        end
        end
    end
    fprintf('ERROR                        NORM                        (in
percent)=%10.4f\n',100*sqrt(numerator/denominator));%error-norm      [in
(%)]
%---

%% DETERMINE THE TEMPERATURE AT SIX UNIFORM POSITIONS PER DIRECTION:
    nctrlX=50; nctrlY=50; nodes=nctrlX*nctrlY;
%---Approximate solution:
    Temp(1:nodes)=0;

    SOL=zeros(nctrlX,nctrlY);
         X=zeros(nctrlX,nctrlY);  Y=zeros(nctrlX,nctrlY);

    Nipmat=zeros(nctrlX,nctrlY,nanchors);
%     Xi=linspace(0,a,nctrlX);
%     Texact(13:16)=Tm*cos(pi*Xi/2/a);

    dy=b/(nctrlY-1); dht=1/(nctrlY-1);
    dx=a/(nctrlX-1); dcs=1/(nctrlX-1);
    nc=0;

    for j=1:nctrlY
        yij=(j-1)*dy; ht=(j-1)*dht;
        for i=1:nctrlX
            xij=(i-1)*dx; cs=(i-1)*dcs;
            Xmesh(i,j)=cs; Ymesh(i,j)=ht;
            Xmesh1(i,j)=xij; Ymesh1(i,j)=yij;
```

```
            nc=nc+1;
             Texact(nc)=Tm*sinh(pi*yij/2/a)                                /
sinh(pi*b/2/a)*cos(pi*xij/2/a);


                    [~, ~,Nip,~,~] = t.blendfunNorm(xij,yij);
                    Temp(nc)=Coef*Nip;
                    Tmesh(i,j)=Temp(nc);
                    Nipmat(i,j,:)=Nip;

                    for k=1:size(t.kVertices)
                    X(i,j)=X(i,j)+Nip(k,1)*XYctrl(1,k);
                    Y(i,j) = Y(i,j) + Nip(k,1) * XYctrl(2,k);
                    SOL(i,j)=SOL(i,j) + Nip(k,1) * Coef(k);

                    end

        end
    end

    figure
    surf(X,Y,SOL)
    colormap('jet');
    shading interp
    xlabel('x')
    ylabel('y')
    xlim([-1 3])
    ylim([-1 12])
    zlim([-10 1200])
    grid off
    figure
    plot(X,Y,'go')
    figure
    hold on
    surf(Xmesh1,Ymesh1,Nipmat(:,:,18))
    colormap('jet');
    shading interp
%
    fprintf('                    \n');
    fprintf('                    \n');
    fprintf('********   R E S U L T S   *******\n');
    fprintf('Node     Temperature      Texact\n');
    for i=1:nodes
        fprintf('%3i    %10.4f      %10.4f\n',i,Temp(i),Texact(i));
    end
```

A.3.2.3 Επίλυση με συναρτήσεις B-spline με δύο εσωτερικούς κόμβους ανά κατεύθυνση (DoF=36)

```
%Test-T-Spline
```

```
%---
%---
clear all
clc;
t = tSpline([],3,[],[],[],[],[]);
%{
31oo32---o33---o34--35oo36
25oo26---o27---o28--29oo30
  ||      |      |      ||
  ||      |      |      ||
19oo20---o21---o22--23oo24
  ||      |      |      ||
  ||      |      |      ||
13oo14---o15---o16--17oo18
  ||      |      |      ||
  ||      |      |      ||
 7oo8----o9--10o----11oo12
  oo-----o-----o------oo
  12     3     4     56

%}
a=3; b=12;
vertices = [ kVertex(0,0,0)       %anchor 1
             kVertex(0,0,0)       %anchor 2
             kVertex(a/3,0,0)       %anchor 3
             kVertex(2*a/3,0,0)       %anchor 4
             kVertex(a,0,0)       %anchor 5
             kVertex(a,0,0)       %anchor 6
             kVertex(0,0,0)       %anchor 7
             kVertex(0,0,0)       %anchor 8
             kVertex(a/3,0,0)       %anchor 9
             kVertex(2*a/3,0,0)       %anchor 10
             kVertex(a,0,0)       %anchor 11
             kVertex(a,0,0)       %anchor 12
             kVertex(0,b/3,0)       %anchor 13
             kVertex(0,b/3,0)       %anchor 14
             kVertex(a/3,b/3,0)       %anchor 15
             kVertex(2*a/3,b/3,0)       %anchor 16
             kVertex(a,b/3,0)       %anchor 17
             kVertex(a,b/3,0)       %anchor 18
             kVertex(0,2*b/3,0)       %anchor 19
             kVertex(0,2*b/3,0)       %anchor 20
             kVertex(a/3,2*b/3,0)       %anchor 21
             kVertex(2*a/3,2*b/3,0)       %anchor 22
             kVertex(a,2*b/3,0)       %anchor 23
             kVertex(a,2*b/3,0)       %anchor 24
             kVertex(0,b,0)       %anchor 25
             kVertex(0,b,0)       %anchor 26
             kVertex(a/3,b,0)       %anchor 27
             kVertex(2*a/3,b,0)       %anchor 28
             kVertex(a,b,0)       %anchor 29
```

```
              kVertex(a,b,0)        %anchor 30
              kVertex(0,b,0)        %anchor 31
              kVertex(0,b,0)        %anchor 32
              kVertex(a/3,b,0)        %anchor 33
              kVertex(2*a/3,b,0)        %anchor 34
              kVertex(a,b,0)        %anchor 35
              kVertex(a,b,0)        %anchor 36
              ];


%% Edges-------------------

vertices(1).cnctRight(vertices(2));
vertices(1).cnctTop(vertices(7));

vertices(2).cnctTop(vertices(8));
vertices(2).cnctRight(vertices(3));
vertices(2).cnctLeft(vertices(1));

vertices(3).cnctTop(vertices(9));
vertices(3).cnctRight(vertices(4));
vertices(3).cnctLeft(vertices(2));

vertices(4).cnctTop(vertices(10));
vertices(4).cnctRight(vertices(5));
vertices(4).cnctLeft(vertices(3));

vertices(5).cnctTop(vertices(11));
vertices(5).cnctRight(vertices(6));
vertices(5).cnctLeft(vertices(4));

vertices(6).cnctTop(vertices(12));
vertices(6).cnctLeft(vertices(5));

vertices(7).cnctTop(vertices(13));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctBottom(vertices(1));

vertices(8).cnctTop(vertices(14));
vertices(8).cnctRight(vertices(9));
vertices(8).cnctBottom(vertices(2));
vertices(8).cnctLeft(vertices(7));

vertices(9).cnctTop(vertices(15));
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(3));
vertices(9).cnctLeft(vertices(8));

vertices(10).cnctTop(vertices(16));
vertices(10).cnctRight(vertices(11));
```

```
vertices(10).cnctBottom(vertices(4));
vertices(10).cnctLeft(vertices(9));

vertices(11).cnctTop(vertices(17));
vertices(11).cnctRight(vertices(12));
vertices(11).cnctBottom(vertices(5));
vertices(11).cnctLeft(vertices(10));

vertices(12).cnctTop(vertices(18));
vertices(12).cnctBottom(vertices(6));
vertices(12).cnctLeft(vertices(11));

vertices(13).cnctTop(vertices(19));
vertices(13).cnctRight(vertices(14));
vertices(13).cnctBottom(vertices(7));

vertices(14).cnctTop(vertices(20));
vertices(14).cnctRight(vertices(15));
vertices(14).cnctBottom(vertices(8));
vertices(14).cnctLeft(vertices(13));

vertices(15).cnctTop(vertices(21));
vertices(15).cnctRight(vertices(16));
vertices(15).cnctBottom(vertices(9));
vertices(15).cnctLeft(vertices(14));

vertices(16).cnctTop(vertices(22));
vertices(16).cnctRight(vertices(17));
vertices(16).cnctBottom(vertices(10));
vertices(16).cnctLeft(vertices(15));

vertices(17).cnctTop(vertices(23));
vertices(17).cnctRight(vertices(18));
vertices(17).cnctBottom(vertices(11));
vertices(17).cnctLeft(vertices(16));

vertices(18).cnctTop(vertices(24));
vertices(18).cnctBottom(vertices(12));
vertices(18).cnctLeft(vertices(17));

vertices(19).cnctTop(vertices(25));
vertices(19).cnctRight(vertices(20));
vertices(19).cnctBottom(vertices(13));

vertices(20).cnctTop(vertices(26));
vertices(20).cnctRight(vertices(21));
vertices(20).cnctBottom(vertices(14));
vertices(20).cnctLeft(vertices(19));
```

```
vertices(21).cnctTop(vertices(27));
vertices(21).cnctRight(vertices(22));
vertices(21).cnctBottom(vertices(15));
vertices(21).cnctLeft(vertices(20));

vertices(22).cnctTop(vertices(28));
vertices(22).cnctRight(vertices(23));
vertices(22).cnctBottom(vertices(16));
vertices(22).cnctLeft(vertices(21));

vertices(23).cnctTop(vertices(29));
vertices(23).cnctRight(vertices(24));
vertices(23).cnctBottom(vertices(17));
vertices(23).cnctLeft(vertices(22));

vertices(24).cnctTop(vertices(30));
vertices(24).cnctBottom(vertices(18));
vertices(24).cnctLeft(vertices(23));

vertices(25).cnctTop(vertices(31));
vertices(25).cnctRight(vertices(26));
vertices(25).cnctBottom(vertices(19));

vertices(26).cnctTop(vertices(32));
vertices(26).cnctRight(vertices(27));
vertices(26).cnctBottom(vertices(20));
vertices(26).cnctLeft(vertices(25));

vertices(27).cnctTop(vertices(33));
vertices(27).cnctRight(vertices(28));
vertices(27).cnctBottom(vertices(21));
vertices(27).cnctLeft(vertices(26));

vertices(28).cnctTop(vertices(34));
vertices(28).cnctRight(vertices(29));
vertices(28).cnctBottom(vertices(22));
vertices(28).cnctLeft(vertices(27));

vertices(29).cnctTop(vertices(35));
vertices(29).cnctRight(vertices(30));
vertices(29).cnctBottom(vertices(23));
vertices(29).cnctLeft(vertices(28));

vertices(30).cnctTop(vertices(36));
vertices(30).cnctBottom(vertices(24));
vertices(30).cnctLeft(vertices(29));

vertices(31).cnctRight(vertices(32));
vertices(31).cnctBottom(vertices(25));
```

```
vertices(32).cnctRight(vertices(33));
vertices(32).cnctBottom(vertices(26));
vertices(32).cnctLeft(vertices(31));

vertices(33).cnctRight(vertices(34));
vertices(33).cnctBottom(vertices(27));
vertices(33).cnctLeft(vertices(32));

vertices(34).cnctRight(vertices(35));
vertices(34).cnctBottom(vertices(28));
vertices(34).cnctLeft(vertices(33));

vertices(35).cnctRight(vertices(36));
vertices(35).cnctBottom(vertices(29));
vertices(35).cnctLeft(vertices(34));

vertices(36).cnctBottom(vertices(30));
vertices(36).cnctLeft(vertices(35));

t.kVertices = vertices;

%t.printPreImagePaper();
t.updateKnotVecs();

sampleCount = 20;
ps = linspace(0,1,sampleCount);
pt = linspace(0,1,sampleCount);


p=3;
nanchors=36;

%% Determine the step we move horizontally and vertically from anchor:
    nc=0;
    if(mod(p,2)==0)
        step=p/2+1;
        ncolumns=2*step;
    else
        step=(p+1)/2;
        ncolumns=2*step+1;
    end

numberOfVertices = size(vertices,1);

for i=1:numberOfVertices
   U_vec(i,1:ncolumns)=vertices(i,1).knotVec(1,:);  %ncolumns=5, for p =
3.
   V_vec(i,1:ncolumns)=vertices(i,1).knotVec(2,:);  %ncolumns=5, for p =
3.
```

```
end
%%

%---
    nx=6; ny=6; %normally: nx=5; ny=6;  5   5
    dx=a/nx; dy=b/ny;
    nc=0;
    for i=1:ny+1
        ht=(i-1)*dy;
        for j=1:nx+1
            nc=nc+1;
            cs=(j-1)*dx;
    [~,~,Nip,dNip_dx,dNip_dy]=t.blendfunNorm(cs,ht);
            Amat(nc,1:nanchors)=Nip;
            RhsX(nc)=cs; %rhs in eq.(1)
            RhsY(nc)=ht; %rhs in eq.(2)
        end
    end
%     Amat
% RhsX
% RhsY
    Xctrl = Amat \ RhsX';   %eq.(1)
    Yctrl = Amat \ RhsY';   %eq.(2)
% figure
% hold on
% plot(xctrl,yctrl,'m+')
hold on
plot(Xctrl,Yctrl,'rx')
% legend('Polytimi','CP','Location','best')
axis equal

    XYctrl(1,1:nanchors)=Xctrl(1:nanchors);
    XYctrl(2,1:nanchors)=Yctrl(1:nanchors);
    XYctrl(3,1:nanchors)=zeros(1,nanchors);

    for i=1:length(vertices)
        vertices(i).cPoint=XYctrl(1:3,i);
    end
    t.kVertices=vertices;
    t.updateKnotVecs;
%---- Ctrl Points

C=zeros(length(vertices),3);
for i=1:length(vertices)
%         Xctrl(i)=vertices(i).cPoint(1);
%         Yctrl(i)=vertices(i).cPoint(2);
        C(i,:)=t.kVertices(i).cPoint(1:3);



end
plot(C(:,1),C(:,2),'ro');
```

```
nctrls = size(C(:,1),1);
dsx=a/50; dsy=b/50;
for i=1:nctrls
    text(C(i,1)+dsx,C(i,2)+dsy,int2str(i),'Color','b');
end
% title('CONTROL POINTS')
axis equal
grid on

% return
%%
%% REPRODUCE THE PATCH AT UNIFORM POSITIONS:
% % %     x = [vertices.s];   %x-coordinates of vertices
% % %     y = [vertices.t];   %y-coordinates of vertices
% % %     xyzctrl=[vertices(:).cPoint];
% % %     xctrl=xyzctrl(1,:); yctrl=xyzctrl(2,:); zctrl=xyzctrl(3,:);
%---
    nx=6; ny=6; %normally: nx=5; ny=6;
    dx=a/nx; dy=b/ny;
    shp=zeros(3,nanchors);
    XYctrl(1,1:nanchors)=Xctrl(1:nanchors);
    XYctrl(2,1:nanchors)=Yctrl(1:nanchors);
%---
    for i=1:ny+1
        ht=(i-1)*dy;
        for j=1:nx+1
            nc=nc+1;
            cs=(j-1)*dx;

                 [~,~,Nip,dNip_dx,dNip_dy] = t.blendfunNorm(cs,ht);

%-----CONSTRUCT JACOBIAN AND ITS INVERSE
nel = nanchors;    %number of control points in the T-mesh
      shp(3,1:nel)=Nip(1:nel);
      shp(1,1:nel)=dNip_dx(1:nel);
      shp(2,1:nel)=dNip_dy(1:nel);
       for ii=1:2
        for jj=1:2
          xs(ii,jj)=0;
           for k=1:nel
            xs(ii,jj)=xs(ii,jj)+XYctrl(ii,k)*shp(jj,k) ;   %Jacobian
           end
        end
       end
% % %
% % %        xs
% % %        break
       %---
      xsj=xs(1,1)*xs(2,2)-xs(1,2)*xs(2,1) ; %detJ
      sx(1,1)=xs(2,2)/xsj ; %Inverse of Jacobian
      sx(2,2)=xs(1,1)/xsj ;
```

```
      sx(1,2)=-xs(1,2)/xsj;
      sx(2,1)=-xs(2,1)/xsj;
%-----FORM GLOBAL DERIVATIVES
      for ii=1:nel
        tp       = shp(1,ii)*sx(1,1)+shp(2,ii)*sx(2,1);
        shp(2,ii) = shp(1,ii)*sx(1,2)+shp(2,ii)*sx(2,2);   %y-derivative
        shp(1,ii)=tp ;                                     %x-derivative
      end
      %---Store detJ:
       DETJacob(i,j)=xsj;

% TELOS
            Xmesh(i,j)=cs; Ymesh(i,j)=ht;
            Xsample(i,j)=Nip'*Xctrl;
            Ysample(i,j)=Nip'*Yctrl;
        end
    end
% figure
%     surf(Xmesh,Ymesh,Xsample)
% figure
%    surf(Xmesh,Ymesh,Ysample)


figure
    plot(Xsample,Ysample,'-o')
    xlabel('X')
    ylabel('Y')

%     return
%%
%%%%STIFFNESS MATRIX (LAUBE FUNS)
%%
%%Calculate the patches:
numberOfPatches=25;
patches=t.patches_calc(numberOfPatches);

% Gauss points and weights:
    [gi,ome] = pgau_rectangle_FULL;
    ngaussx=p+1;   %ceil((1+p^2)/2);
    ngaussy=ngaussx;
 %% Stiffness matrix
    K=zeros(nanchors);  %initialize stiffness matrix
    M=zeros(nanchors);  %initialize stiffness matrix
    area=0;
% % %
% % %         taugaussx=(1+gi(1:ngaussx,ngaussx))/2;  %0<Ugauss<1
% % %         dRipx=
% % %         taugaussy=(1+gi(1:ngaussy,ngaussy))/2;  %0<Vgauss<1
% % %
    for k=1:numberOfPatches
        %Corner points:
```

```
        xA=patches(k,1); yA=patches(k,2);
        xB=patches(k,3); yB=patches(k,4);
        xC=patches(k,5); yC=patches(k,6);
        xD=patches(k,7); yD=patches(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
        taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
        taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);
%                                                [Nip,dNip_dx,dNip_dy]     =
Ripfun(nanchors,U_vec,V_vec,ncolumns,p,...
%                                            taugaussx,taugaussy);


B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y]                          =
t.blendfun(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:);  %matrix  4x4x16  (4x4  gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end

 fprintf('sumB_mat=%3i\n',sum(B_mat(1,2,:)));

        for igx=1:ngaussy

            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
        fprintf('Patch=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

        for i=1:nanchors
                    dNix=dB_xmat(igx,igy,i);
                    dNiy=dB_ymat(igx,igy,i);
                    for j=1:nanchors
                        dNjx=dB_xmat(igx,igy,j);
                        dNjy=dB_ymat(igx,igy,j);
                        K(i,j)=K(i,j)+(dNix*dNjx+dNiy*dNjy)*...
```

```
detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                        M(i,j)=M(i,j)+B_mat(igx,igy,i)                 *
B_mat(igx,igy,j)*...

detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                    end
         end



            end
        end
    end
    end
    area

    K;
%---

try chol(M);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end
sumM=0; sumK=0;

    for i=1:nanchors
        for j=1:nanchors
            sumK=sumK+K(i,j);
            sumM=sumM+M(i,j);
        end
    end     %end-of-IF detJ~= 0.
    fprintf('SumKij=%12.5e  SumMij=%12.5e\n',sumK,sumM);

%     return
%}

%% IMPOSE BOUNDARY CONDITIONS
%{
 D                    C
31oo32---o33---o34--35oo36
25oo26---o27---o28--29oo30
  ||      |      |      ||
  ||      |      |      ||
19oo20---o21---o22--23oo24
  ||      |      |      ||
  ||      |      |      ||
13oo14---o15---o16--17oo18
  ||      |      |      ||
  ||      |      |      ||
 7oo8----o9--10o----11oo12
```

```
  oo-----o-----o------oo
  12     3     4     56
A                        B
%}
    BC(1:36)=0; %initially all DOFs are unrestrained
    BC(1:6)=1; BC(12:6:30)=1;   BC(31:36)=1;
    fi(1:6)=0; fi(12:6:30)=0;  fi(31:36)=0;
%---Find the coefficients along the side DC using equally-spaced spans:
    knots_DC = [0,0,0,0,a/3, 2*a/3,a,a,a,a];
    xtest_DC=linspace(0,3,6);
    Utest_DC=1000*cos(pi*xtest_DC/6);
    nstep=1; %We need only two terms(Nip and first derivative).
    p=3;
    colmatTEST_DC=spcol(knots_DC,p+1,brk2knt(xtest_DC,nstep));  %1  term
(Nip-only)
    i1y=size(colmatTEST_DC,1);
    BasisTEST_DC = colmatTEST_DC(1:nstep:i1y,:);
%---Using the above matrix, determine the control points:
    XcontrolP_DC = BasisTEST_DC \ xtest_DC';
%---Using the above matrix, determine the coefficients:
    Coef_DC = BasisTEST_DC \ Utest_DC';
    fi(31:36)=Coef_DC;
%---Quality of representing the BC along DC:
    nseg=100;
    tauseg=linspace(0,3,nseg+1);
    colmatseg=spcol(knots_DC,p+1,brk2knt(tauseg,nstep));  %1  term  (Nip-
only)
    i1y=size(colmatseg,1);
    Basisseg = colmatseg(1:nstep:i1y,:);
    Useg=Basisseg*Coef_DC;
%
    figure
    plot(tauseg,Useg,'bo')
    hold on
    plot(tauseg,1000*cos(pi*tauseg/6),'r','Linewidth',2)
    xlabel('X')
    ylabel('T(x)')
    title('CALCULATED versus EXACT along DC')
    legend('Approximation','Exact','Location','best')
%---Arrange knowns and unknowns:
    free_dofs = find(BC == 0);
    fixed_dofs= find(BC == 1);
%

%     size(K)
%     size(free_dofs)
%     size(fixed_dofs)
%     size(K(free_dofs,fixed_dofs))
%     pause
%
    rhs = -K(free_dofs,fixed_dofs)*fi(fixed_dofs)';
```

```
    Coef(1:36)=0;
    Coef(free_dofs) = K(free_dofs,free_dofs) \ rhs;
%---Complete the BC in a single  Coef:
    Coef(fixed_dofs) = 0;
    Coef(31:36)=Coef_DC;
%-----------------------------------------------------------------------
----

%% Error norm (L2): >>>
    numerator=0;
    denominator=0;
    area=0;
    Tm=1000; a=3; b=12;
    ngaussx=p+1;
    ngaussy=p+1;
    for k=1:numberOfPatches
        %Corner points:
        xA=patches(k,1); yA=patches(k,2);
        xB=patches(k,3); yB=patches(k,4);
        xC=patches(k,5); yC=patches(k,6);
        xD=patches(k,7); yD=patches(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
            taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
            taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);

B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y]                          =
t.blendfunNorm(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:);  %matrix  4x4x16  (4x4  gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end
%          %
        for igx=1:ngaussy
            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                xgpt = taugaussx(igy); ygpt = taugaussy(igx);
```

```
                 Tex          =          Tm*sinh(pi*ygpt/2/a)          /
sinh(pi*b/2/a)*cos(pi*xgpt/2/a);
               T=0;
               for i=1:nanchors
                  T = T +  B_mat(igx,igy,i)*Coef(i);
%                 T = T +  Nip(i,igx,igy)*Coef(i);
               end
           numerator=numerator+(T-
Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);

denominator=denominator+(Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);
            end
        end
        end
    end
    fprintf('ERROR                        NORM                        (in
percent)=%10.4f\n',100*sqrt(numerator/denominator));%error-norm      [in
(%)]
%---

%% DETERMINE THE TEMPERATURE AT SIX UNIFORM POSITIONS PER DIRECTION:
    nctrlX=6; nctrlY=6; nodes=nctrlX*nctrlY;
%---Approximate solution:
    Temp(1:nodes)=0;
%     Xi=linspace(0,a,nctrlX);
%     Texact(13:16)=Tm*cos(pi*Xi/2/a);

    dy=b/(nctrlY-1); dht=1/(nctrlY-1);
    dx=a/(nctrlX-1); dcs=1/(nctrlX-1);
    nc=0;
    for j=1:nctrlY
        yij=(j-1)*dy; ht=(j-1)*dht;
        for i=1:nctrlX
            xij=(i-1)*dx; cs=(i-1)*dcs;
            nc=nc+1;
             Texact(nc)=Tm*sinh(pi*yij/2/a)                            /
sinh(pi*b/2/a)*cos(pi*xij/2/a);

                    [~, ~,Nip,~,~] = t.blendfunNorm(xij,yij);
                    Temp(nc)=Coef*Nip;
        end
    end
%
    fprintf('                    \n');
    fprintf('                    \n');
    fprintf('********   R E S U L T S   *******\n');
    fprintf('Node     Temperature      Texact\n');
    for i=1:nodes
        fprintf('%3i    %10.4f      %10.4f\n',i,Temp(i),Texact(i));
    end
```

A.3.2.4 Επίλυση με συναρτήσεις T-spline (DoF=32)

```
%Test-T-Spline
%---
clear all
clc;
t = tSpline([],3,[],[],[],[],[]);

%{
27oo28--29o--30o---------31oo32
21oo22--23o--24o---------25oo26
  ||      |    |           ||
  ||      |    |           ||
17oo18--19o--20o           ||
  ||      |    |           ||
  ||      |    |           ||
11oo12--13o--14o---------15oo16
  ||           |           ||
  ||           |           ||
  ||           |           ||
  ||           |           ||
 6oo7---------8o----------9oo10
  oo-----------o-----------oo
  12           3           45

%}
a=3; b=12;
vertices = [ kVertex(0,0,0)       %anchor 1
             kVertex(0,0,0)       %anchor 2
             kVertex(a/2,0,0)     %anchor 3
             kVertex(a,0,0)       %anchor 4
             kVertex(a,0,0)       %anchor 5
             kVertex(0,0,0)       %anchor 6
             kVertex(0,0,0)       %anchor 7
             kVertex(a/2,0,0)     %anchor 8
             kVertex(a,0,0)       %anchor 9
             kVertex(a,0,0)       %anchor 10
             kVertex(0,b/2,0)       %anchor 11
             kVertex(0,b/2,0)       %anchor 12
             kVertex(a/4,b/2,0)     %anchor 13
             kVertex(a/2,b/2,0)     %anchor 14
             kVertex(a,b/2,0)       %anchor 15
             kVertex(a,b/2,0)       %anchor 16
             kVertex(0,3*b/4,0)       %anchor 17
             kVertex(0,3*b/4,0)       %anchor 18
             kVertex(a/4,3*b/4,0)     %anchor 19
             kVertex(a/2,3*b/4,0)     %anchor 20
             kVertex(0,b,0)      %anchor 21
             kVertex(0,b,0)      %anchor 22
             kVertex(a/4,b,0)    %anchor 23
             kVertex(a/2,b,0)    %anchor 24
             kVertex(a,b,0)      %anchor 25
```

```
              kVertex(a,b,0)       %anchor 26
              kVertex(0,b,0)       %anchor 27
              kVertex(0,b,0)       %anchor 28
              kVertex(a/4,b,0)     %anchor 29
              kVertex(a/2,b,0)     %anchor 30
              kVertex(a,b,0)       %anchor 31
              kVertex(a,b,0)];     %anchor 32


%% Edges-------------------

vertices(1).cnctRight(vertices(2));
vertices(1).cnctTop(vertices(6));

vertices(2).cnctTop(vertices(7));
vertices(2).cnctRight(vertices(3));
vertices(2).cnctLeft(vertices(1));

vertices(3).cnctTop(vertices(8));
vertices(3).cnctRight(vertices(4));
vertices(3).cnctLeft(vertices(2));

vertices(4).cnctTop(vertices(9));
vertices(4).cnctRight(vertices(5));
vertices(4).cnctLeft(vertices(3));

vertices(5).cnctTop(vertices(10));
vertices(5).cnctLeft(vertices(4));

vertices(6).cnctTop(vertices(11));
vertices(6).cnctRight(vertices(7));
vertices(6).cnctBottom(vertices(1));

vertices(7).cnctTop(vertices(12));
vertices(7).cnctRight(vertices(8));
vertices(7).cnctBottom(vertices(2));
vertices(7).cnctLeft(vertices(6));

vertices(8).cnctTop(vertices(14));
vertices(8).cnctRight(vertices(9));
vertices(8).cnctBottom(vertices(3));
vertices(8).cnctLeft(vertices(7));

vertices(9).cnctTop(vertices(15));
vertices(9).cnctRight(vertices(10));
vertices(9).cnctBottom(vertices(4));
vertices(9).cnctLeft(vertices(8));

vertices(10).cnctTop(vertices(16));
vertices(10).cnctBottom(vertices(5));
vertices(10).cnctLeft(vertices(9));
```

```
vertices(11).cnctTop(vertices(17));
vertices(11).cnctRight(vertices(12));
vertices(11).cnctBottom(vertices(6));

vertices(12).cnctTop(vertices(18));
vertices(12).cnctRight(vertices(13));
vertices(12).cnctBottom(vertices(7));
vertices(12).cnctLeft(vertices(11));

vertices(13).cnctTop(vertices(19));
vertices(13).cnctRight(vertices(14));
vertices(13).cnctLeft(vertices(12));

vertices(14).cnctTop(vertices(20));
vertices(14).cnctRight(vertices(15));
vertices(14).cnctBottom(vertices(8));
vertices(14).cnctLeft(vertices(13));

vertices(15).cnctTop(vertices(25));
vertices(15).cnctRight(vertices(16));
vertices(15).cnctBottom(vertices(9));
vertices(15).cnctLeft(vertices(14));

vertices(16).cnctTop(vertices(26));
vertices(16).cnctBottom(vertices(10));
vertices(16).cnctLeft(vertices(15));

vertices(17).cnctTop(vertices(21));
vertices(17).cnctRight(vertices(18));
vertices(17).cnctBottom(vertices(11));

vertices(18).cnctTop(vertices(22));
vertices(18).cnctRight(vertices(19));
vertices(18).cnctBottom(vertices(12));
vertices(18).cnctLeft(vertices(17));

vertices(19).cnctTop(vertices(23));
vertices(19).cnctRight(vertices(20));
vertices(19).cnctBottom(vertices(13));
vertices(19).cnctLeft(vertices(18));

vertices(20).cnctTop(vertices(24));
vertices(20).cnctBottom(vertices(14));
vertices(20).cnctLeft(vertices(19));

vertices(21).cnctTop(vertices(27));
vertices(21).cnctRight(vertices(22));
vertices(21).cnctBottom(vertices(17));

vertices(22).cnctTop(vertices(28));
vertices(22).cnctRight(vertices(23));
```

```
vertices(22).cnctBottom(vertices(18));
vertices(22).cnctLeft(vertices(21));

vertices(23).cnctTop(vertices(29));
vertices(23).cnctRight(vertices(24));
vertices(23).cnctBottom(vertices(19));
vertices(23).cnctLeft(vertices(22));

vertices(24).cnctTop(vertices(30));
vertices(24).cnctRight(vertices(25));
vertices(24).cnctBottom(vertices(20));
vertices(24).cnctLeft(vertices(23));

vertices(25).cnctTop(vertices(31));
vertices(25).cnctRight(vertices(26));
vertices(25).cnctBottom(vertices(15));
vertices(25).cnctLeft(vertices(24));

vertices(26).cnctTop(vertices(32));
vertices(26).cnctBottom(vertices(16));
vertices(26).cnctLeft(vertices(25));

vertices(27).cnctRight(vertices(28));
vertices(27).cnctBottom(vertices(21));

vertices(28).cnctRight(vertices(29));
vertices(28).cnctBottom(vertices(22));
vertices(28).cnctLeft(vertices(27));

vertices(29).cnctRight(vertices(30));
vertices(29).cnctBottom(vertices(23));
vertices(29).cnctLeft(vertices(28));

vertices(30).cnctRight(vertices(31));
vertices(30).cnctBottom(vertices(24));
vertices(30).cnctLeft(vertices(29));

vertices(31).cnctRight(vertices(32));
vertices(31).cnctBottom(vertices(25));
vertices(31).cnctLeft(vertices(30));

vertices(32).cnctBottom(vertices(26));
vertices(32).cnctLeft(vertices(31));

t.kVertices = vertices;


% figure
% t.printPreImagePaper();

t.updateKnotVecs();
```

```
p=3;
nanchors=32;

% return
%% Determine the step we move horizontally and vertically from anchor:
    nc=0;
    if(mod(p,2)==0)
        step=p/2+1;
        ncolumns=2*step;
    else
        step=(p+1)/2;
        ncolumns=2*step+1;
    end

numberOfVertices = size(vertices,1);

for i=1:numberOfVertices
   U_vec(i,1:ncolumns)=vertices(i,1).knotVec(1,:);  %ncolumns=5, for p
= 3.
   V_vec(i,1:ncolumns)=vertices(i,1).knotVec(2,:);  %ncolumns=5, for p
= 3.
end

%%
%% DETERMINE CONTROL POINTS SO AS TO PRESERVE LINEAR MAPPING: BLENDFUN
%---
    nx=7; ny=7; %normally: nx=5; ny=6;  5   5
    dx=a/nx; dy=b/ny;
    nc=0;
    for i=1:ny+1
        ht=(i-1)*dy;
        for j=1:nx+1
            nc=nc+1;
            cs=(j-1)*dx;
%                                       cs,ht);
            [~,~,Bip,~,~] = t.blendfunNorm(cs,ht);
            Amat(nc,1:nanchors)=Bip;
            RhsX(nc)=cs; %rhs in eq.(1)
            RhsY(nc)=ht; %rhs in eq.(2)
        end
    end
%     Amat2=Amat
% RhsX2=RhsX
% RhsY2=RhsY
    Xctrl = Amat \ RhsX';   %eq.(1)
    Yctrl = Amat \ RhsY';   %eq.(2)
figure
% hold on
% plot(xctrl,yctrl,'m+')
hold on
plot(Xctrl,Yctrl,'rx')
```

```
% legend('Polytimi','CP','Location','best')
axis equal

    XYctrl(1,1:nanchors)=Xctrl(1:nanchors);
    XYctrl(2,1:nanchors)=Yctrl(1:nanchors);
    XYctrl(3,1:nanchors)=zeros(1,nanchors);

    for i=1:length(vertices)
        vertices(i).cPoint=XYctrl(1:3,i);
    end
    t.kVertices=vertices;
    t.updateKnotVecs();
%---- Ctrl Points

C=zeros(length(vertices),3);
for i=1:length(vertices)
%         Xctrl(i)=vertices(i).cPoint(1);
%         Yctrl(i)=vertices(i).cPoint(2);
        C(i,:)=t.kVertices(i).cPoint(1:3);
%         ST(i,:)=[t.kVertices(i)];


end
plot(C(:,1),C(:,2),'ro');
nctrls = size(C(:,1),1);
dsx=a/50; dsy=b/50;
for i=1:nctrls
    text(C(i,1)+dsx,C(i,2)+dsy,int2str(i),'Color','b');
end
% title('CONTROL POINTS')
axis equal
grid on

% return
%% REPRODUCE THE PATCH AT UNIFORM POSITIONS:
% % %     x = [vertices.s];   %x-coordinates of vertices
% % %     y = [vertices.t];   %y-coordinates of vertices
% % %     xyzctrl=[vertices(:).cPoint];
% % %     xctrl=xyzctrl(1,:); yctrl=xyzctrl(2,:); zctrl=xyzctrl(3,:);
%---
    nx=10; ny=10; %normally: nx=5; ny=6;
    dx=a/nx; dy=b/ny;
    shp=zeros(3,nanchors);
    XYctrl(1,1:nanchors)=Xctrl(1:nanchors);
    XYctrl(2,1:nanchors)=Yctrl(1:nanchors);
%---
    for i=1:ny+1
        ht=(i-1)*dy;
        for j=1:nx+1
            nc=nc+1;
            cs=(j-1)*dx;
```

```
            [~,~,Nip,dNip_dx,dNip_dy] = t.blendfunNorm(cs,ht);

%-----CONSTRUCT JACOBIAN AND ITS INVERSE
nel = nanchors;    %number of control points in the T-mesh
      shp(3,1:nel)=Nip(1:nel);
      shp(1,1:nel)=dNip_dx(1:nel);
      shp(2,1:nel)=dNip_dy(1:nel);
       for ii=1:2
        for jj=1:2
          xs(ii,jj)=0;
           for k=1:nel
            xs(ii,jj)=xs(ii,jj)+XYctrl(ii,k)*shp(jj,k) ;   %Jacobian
           end
        end
       end
% % %
% % %        xs
% % %        break
       %---
      xsj=xs(1,1)*xs(2,2)-xs(1,2)*xs(2,1) ; %detJ
      sx(1,1)=xs(2,2)/xsj ; %Inverse of Jacobian
      sx(2,2)=xs(1,1)/xsj ;
      sx(1,2)=-xs(1,2)/xsj;
      sx(2,1)=-xs(2,1)/xsj;
%-----FORM GLOBAL DERIVATIVES
      for ii=1:nel
        tp       = shp(1,ii)*sx(1,1)+shp(2,ii)*sx(2,1);
        shp(2,ii) = shp(1,ii)*sx(1,2)+shp(2,ii)*sx(2,2);   %y-
derivative
        shp(1,ii)=tp ;                                    %x-derivative
      end
      %---Store detJ:
       DETJacob(i,j)=xsj;
% TELOS
            Xmesh(i,j)=cs; Ymesh(i,j)=ht;
            Xsample(i,j)=Nip'*Xctrl;
            Ysample(i,j)=Nip'*Yctrl;
        end
    end
% figure
%     surf(Xmesh,Ymesh,Xsample)
% figure
%    surf(Xmesh,Ymesh,Ysample)
figure
    plot(Xsample,Ysample,'-o')
    xlabel('X')
    ylabel('Y')
%%
%%STIFFNESS MATRIX (LAUBE FUNS)

%%Calculate the patches:
numberOfPatches=21;
```

```
patches=t.patches_calc(numberOfPatches);

%%Calculate the elements:
t.fVert_last
fVertices=t.fVertices;

fVertices(1).cnctRight(vertices(8));
vertices(8).cnctLeft(fVertices(1));
fVertices(1).cnctLeft(vertices(7));
vertices(7).cnctRight(fVertices(1));

fVertices(2).cnctRight(vertices(3));
vertices(3).cnctLeft(fVertices(2));
fVertices(2).cnctLeft(vertices(2));
vertices(2).cnctRight(fVertices(2));

fVertices(3).cnctBottom(vertices(15));
vertices(15).cnctTop(fVertices(3));
fVertices(3).cnctTop(vertices(25));
vertices(25).cnctBottom(fVertices(3));

fVertices(4).cnctBottom(vertices(16));
vertices(16).cnctTop(fVertices(4));
fVertices(4).cnctTop(vertices(26));
vertices(26).cnctBottom(fVertices(4));

noelmnts=9;
elmnts_pts=element_calc(t,noelmnts);
% Gauss points and weights:
    [gi,ome] = pgau_rectangle_FULL;
    ngaussx=p+1;   %ceil((1+p^2)/2);
    ngaussy=ngaussx;
 %% Stiffness matrix
    K=zeros(nanchors);  %initialize stiffness matrix
    M=zeros(nanchors);  %initialize stiffness matrix
    area=0;
% % %
% % %         taugaussx=(1+gi(1:ngaussx,ngaussx))/2;  %0<Ugauss<1
% % %         dRipx=
% % %         taugaussy=(1+gi(1:ngaussy,ngaussy))/2;  %0<Vgauss<1
% % %
    for k=1:noelmnts
        %Corner points:
        xA=elmnts_pts(k,1); yA=elmnts_pts(k,2);
        xB=elmnts_pts(k,3); yB=elmnts_pts(k,4);
        xC=elmnts_pts(k,5); yC=elmnts_pts(k,6);
        xD=elmnts_pts(k,7); yD=elmnts_pts(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
```

```
        %
        if(detJ ~= 0)
        taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
        taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);


B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y] =
t.blendfunNorm(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:); %matrix 4x4x16 (4x4 gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end

 fprintf('sumB_mat=%3i\n',sum(B_mat(1,2,:)));

        for igx=1:ngaussy

            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
        fprintf('Element=%2i  igx=%2i  igy=%2i\n',k,igx,igy);

        for i=1:nanchors
                    dNix=dB_xmat(igx,igy,i);
                    dNiy=dB_ymat(igx,igy,i);
                    for j=1:nanchors
                        dNjx=dB_xmat(igx,igy,j);
                        dNjy=dB_ymat(igx,igy,j);
                        K(i,j)=K(i,j)+(dNix*dNjx+dNiy*dNjy)*...

detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                        M(i,j)=M(i,j)+B_mat(igx,igy,i) *
B_mat(igx,igy,j)*...

detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                    end
         end


            end
        end
    end
    end
```

```
    area

    K;
%---

try chol(M);
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end
sumM=0; sumK=0;

    for i=1:nanchors
        for j=1:nanchors
            sumK=sumK+K(i,j);
            sumM=sumM+M(i,j);
        end
    end     %end-of-IF detJ~= 0.
    fprintf('SumKij=%12.5e  SumMij=%12.5e\n',sumK,sumM);

%     break
%}

%% IMPOSE BOUNDARY CONDITIONS
%{
 D                     C
27oo28--29o--30o---------31oo32
21oo22--23o--24o---------25oo26
  ||      |    |           ||
  ||      |    |           ||
17oo18--19o--20o           ||
  ||      |    |           ||
  ||      |    |           ||
11oo12--13o--14o---------15oo16
  ||           |           ||
  ||           |           ||
  ||           |           ||
  ||           |           ||
 6oo7---------8o----------9oo10
  oo-----------o-----------oo
A 12           3           45 B
%}
    BC(1:32)=0; %initially all DOFs are unrestrained
    BC(1:5)=1; BC(10)=1; BC(16)=1; BC(26)=1;   BC(27:32)=1;
    fi(1:5)=0; fi(10)=0; fi(16)=0; fi(26)=0;   fi(27:32)=0;
%---Find the coefficients along the side DC using equally-spaced spans:
    knots_DC = [0,0,0,0,0.75,1.5,3,3,3,3];
    xtest_DC=linspace(0,3,6);
    Utest_DC=1000*cos(pi*xtest_DC/6);
    nstep=1; %We need only two terms(Nip and first derivative).
    p=3;
```

```
    colmatTEST_DC=spcol(knots_DC,p+1,brk2knt(xtest_DC,nstep)); %1 term
(Nip-only)
    i1y=size(colmatTEST_DC,1);
    BasisTEST_DC = colmatTEST_DC(1:nstep:i1y,:);
%---Using the above matrix, determine the control points:
    XcontrolP_DC = BasisTEST_DC \ xtest_DC';
%---Using the above matrix, determine the coefficients:
    Coef_DC = BasisTEST_DC \ Utest_DC';
    fi(27:32)=Coef_DC;
%---Quality of representing the BC along DC:
    nseg=100;
    tauseg=linspace(0,3,nseg+1);
    colmatseg=spcol(knots_DC,p+1,brk2knt(tauseg,nstep)); %1 term (Nip-
only)
    i1y=size(colmatseg,1);
    Basisseg = colmatseg(1:nstep:i1y,:);
    Useg=Basisseg*Coef_DC;
%
    figure
    plot(tauseg,Useg,'bo')
    hold on
    plot(tauseg,1000*cos(pi*tauseg/6),'r','Linewidth',2)
    xlabel('X')
    ylabel('T(x)')
    title('CALCULATED versus EXACT along DC')
    legend('Approximation','Exact','Location','best')
%---Arrange knowns and unknowns:
    free_dofs = find(BC == 0);
    fixed_dofs= find(BC == 1);
%

%
    rhs = -K(free_dofs,fixed_dofs)*fi(fixed_dofs)';
    Coef(1:32)=0;
    Coef(free_dofs) = K(free_dofs,free_dofs) \ rhs;
%---Complete the BC in a single Coef:
    Coef(fixed_dofs) = 0;
    Coef(27:32)=Coef_DC;
%-----------------------------------------------------------------------
----

%% Error norm (L2): >>>
    numerator=0;
    denominator=0;
    area=0;
    Tm=1000; a=3; b=12;
    %% NEW-STYLE (Provatidis: Friday morning 6:47, 7/2/2020):
    ngaussx=p+1;
    ngaussy=p+1;
    for k=1:noelmnts
        %Corner points:
        xA=elmnts_pts(k,1); yA=elmnts_pts(k,2);
```

```
        xB=elmnts_pts(k,3); yB=elmnts_pts(k,4);
        xC=elmnts_pts(k,5); yC=elmnts_pts(k,6);
        xD=elmnts_pts(k,7); yD=elmnts_pts(k,8);
        %Centroid:
        xc=0.25*(xA+xB+xC+xD); yc=0.25*(yA+yB+yC+yD);
        %Jacobian:
        detJ = (xB-xA)*(yC-yB)/4;
        %
        if(detJ ~= 0)
            taugaussx=xc+(xB-xA)/2*gi(1:ngaussx,ngaussx);
            taugaussy=yc+(yC-yB)/2*gi(1:ngaussy,ngaussy);

B_mat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_xmat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));

dB_ymat=zeros(length(taugaussx),length(taugaussy),size(vertices,1));
        for i=1:size(taugaussx,1)
            for j=1:size(taugaussy,1)
                   [BS,BT,Bip,dBip_x,dBip_y] =
t.blendfunNorm(taugaussx(i),taugaussy(j));
                    B_mat(j,i,:)=Bip(:); %matrix 4x4x16 (4x4 gaussian
points// 16 blending functions)
                    dB_xmat(j,i,:)=dBip_x(:);
                    dB_ymat(j,i,:)=dBip_y(:);

            end
        end
        for igx=1:ngaussy
            for igy=1:ngaussx
                area = area + detJ*ome(igy,ngaussx)*ome(igx,ngaussy);
                xgpt = taugaussx(igy); ygpt = taugaussy(igx);
                Tex = Tm*sinh(pi*ygpt/2/a) /
sinh(pi*b/2/a)*cos(pi*xgpt/2/a);
               T=0;
               for i=1:nanchors
                  T = T +  B_mat(igx,igy,i)*Coef(i);
               end
           numerator=numerator+(T-
Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);

denominator=denominator+(Tex)^2*ome(igx,ngaussy)*ome(igy,ngaussx);
            end
        end
        end
    end
    fprintf('ERROR NORM (in
percent)=%10.4f\n',100*sqrt(numerator/denominator)*detJ);%error-norm
[in (%)]
%---
% return
```

```
%% DETERMINE THE TEMPERATURE AT SIX UNIFORM POSITIONS PER DIRECTION:
    nctrlX=6; nctrlY=6; nodes=nctrlX*nctrlY;
%---Approximate solution:
    Temp(1:nodes)=0;
%     Xi=linspace(0,a,nctrlX);
%     Texact(13:16)=Tm*cos(pi*Xi/2/a);

    dy=b/(nctrlY-1); dht=1/(nctrlY-1);
    dx=a/(nctrlX-1); dcs=1/(nctrlX-1);
    nc=0;
    for j=1:nctrlY
        yij=(j-1)*dy; ht=(j-1)*dht;
        for i=1:nctrlX
            xij=(i-1)*dx; cs=(i-1)*dcs;
            nc=nc+1;
             Texact(nc)=Tm*sinh(pi*yij/2/a) /
sinh(pi*b/2/a)*cos(pi*xij/2/a);

                    [~, ~,Nip,~,~] = t.blendfunNorm(xij,yij);
                     Temp(nc)=Coef*Nip;


        end
    end
%
    fprintf('                    \n');
    fprintf('                    \n');
    fprintf('********   R E S U L T S   *******\n');
    fprintf('Node     Temperature      Texact\n');
    for i=1:nodes
        fprintf('%3i    %10.4f      %10.4f\n',i,Temp(i),Texact(i));
    end

%     break

return

%% Impose linear mapping along the bottom side (AB):
%  Note that this side is INDEPENDENT of all other data(!)
    knots=[0,0,0,0,4,6,8,12,12,12,12];
% Determine the six(6) control point which impose linear mapping:
% First we choose seven test points at equal spaces
    tauTEST=linspace(0,12,7);
    nstep=1; %We need only two terms(Nip and first derivative).
    p=3;
    colmatTEST=spcol(knots,p+1,brk2knt(tauTEST,nstep)); %1 term (Nip-
only)
    i1y=size(colmatTEST,1);
    BasisTEST = colmatTEST(1:nstep:i1y,:);
%---Using the above matrix, determine the control points:
    XcontrolP = BasisTEST \ tauTEST'
```

```
%% ALTERNATIVE WAY TO DETERMINE THE CTRLs BY KNOT INSERTION AT THE
ENDS:
%% Edge AB
    knotsAB=[0,0,0,0, 3,3,3,3];  %8 knots (cubic Bézier)
    ctrls = [0, 1, 2, 3];    %control points (uniform cubic Bézier)
%--- Make the B-form (de Boor) spline STRUCTURE:
    sp=spmak(knotsAB,ctrls);       %2x3 STRUCTURE [B-form (de Boor)]
%--- FIRST REFINEMENT (Knot Insertion)
     addpts=[1, 2];    %additional points, same as the T-mesh knots
     sp2AB=fnrfn(sp,addpts);    %2x4 new structure (refined)
 %--- SHOW THE NEW (UPDATED) CONTROL POINTS::
    sp2AB.coefs

    [XcontrolP sp2AB.coefs']

%% Edge BC:
    knotsBC = [0,0,0,0, 12,12,12,12];
    ctrls = [0,4,8,12];
    sp=spmak(knotsBC,ctrls);       %2x3 STRUCTURE [B-form (de Boor)]
    addpts = [4, 8];
    sp2BC=fnrfn(sp,addpts);       %2x4 new structure (refined)
    sp2BC.coefs'

%% Edge DC:
    knotsDC = [0,0,0,0, 3,3,3,3];
    ctrls = [0,1,2,3];
    sp=spmak(knotsDC,ctrls);       %2x3 STRUCTURE [B-form (de Boor)]
    addpts = [1,2];
    sp2DC=fnrfn(sp,addpts);       %2x4 new structure (refined)
    sp2DC.coefs'


%% Edge AD:
    knotsAD = [0,0,0,0, 12,12,12,12];
    ctrls = [0,4,8,12];
    sp=spmak(knotsAD,ctrls);       %2x3 STRUCTURE [B-form (de Boor)]
    addpts = [6, 9];    %inserted knots
    sp2AD=fnrfn(sp,addpts);       %2x4 new structure (refined)
    sp2AD.coefs'

%% NEXT TO DO:
% Evaluation using LAUBE-DIMITRIOU κυρίως για τα διαγώνια σημεία
ελέγχου
```

A.4 Συναρτήσεις εντός του υπολογιστικού πακέτου

A.4.1 Συνάρτηση υπολογισμού των πολυωνυμικών συναρτήσεων T-spline

Με την εν λόγω συνάρτηση υπολογίζονται οι πολυωνυμικές συναρτήσεις T-spline σε κάθε κατεύθυνση διδιάστατες πολυωνυμικές συναρτήσεις T-spline, καθώς και οι παράγωγοι τους ως προς τις παραμετρικές συντεταγμένες στις θέσεις `s` και `t` του παραμετρικού χωρίου.

Μεταβλητές εισόδου: `obj, s, t`

Μεταβλητές εξόδου: `BS,BT, Nip, dNip_x, dNip_y`

```
function [BS, BT,Nip,dNip_x,dNip_y] = blendfun(obj, s, t)
            Nip=zeros(size(obj.kVertices,1),1);
            dNip_x=zeros(size(obj.kVertices,1),1);
            dNip_y=zeros(size(obj.kVertices,1),1);
            BS=zeros(size(obj.kVertices,1),1);
            BU=zeros(size(obj.kVertices,1),1);
            BT=zeros(size(obj.kVertices,1),1);
            BV=zeros(size(obj.kVertices,1),1);
            for i=1:size(obj.kVertices,1)
                knotspanS = 1;
                knotspanT = 1;

                bs = basisFunTSpl(s, obj.kVertices(i).knotVec(1,:),
obj.deg, knotspanS);   %Basis functions
                bt = basisFunTSpl(t, obj.kVertices(i).knotVec(2,:),
obj.deg, knotspanT);   %Basis functions
                dN_s =
derRecur(s,obj.kVertices(i).knotVec(1,:),obj.deg, knotspanS);
%Derivative over x
                dN_t =
derRecur(t,obj.kVertices(i).knotVec(2,:),obj.deg, knotspanT);
%Derivative over y
                BS(i)=bs;
                BT(i)=bt;

                B = sum(bs)*sum(bt); %Blending function
                Nip(i)=B;
                dNip_x(i)=dN_s*bt; %dB/dx
                dNip_y(i)=dN_t*bs; %dB/dy


            end
        end
```

A.4.2 Συνάρτηση υπολογισμού των ρητών συναρτήσεων T-spline χωρίς βάρος

Με την εν λόγω συνάρτηση υπολογίζονται οι πολυωνυμικές συναρτήσεις T-spline σε κάθε κατεύθυνση, οι ρητές διδιάστατες συναρτήσεις T-spline, χωρίς χρήση συντελεστών βάρους και οι μερικές παράγωγοι τους στα παραμετρικά σημεία `s` και `t`.

Μεταβλητές εισόδου: `obj, s, t`

Μεταβλητές εξόδου: `BS, BT,Bip,dBip_x,dBip_y`

```
function [BS, BT,Bip,dBip_x,dBip_y] = blendfunNorm(obj, s, t)
            Bip=zeros(size(obj.kVertices,1),1);
            dBip_x=zeros(size(obj.kVertices,1),1);
            dBip_y=zeros(size(obj.kVertices,1),1);
            BS=zeros(size(obj.kVertices,1),1);
            BT=zeros(size(obj.kVertices,1),1);
            for i=1:size(obj.kVertices,1)
                knotspanS = 1;
                knotspanT = 1;

                bs = basisFunTSpl(s, obj.kVertices(i).knotVec(1,:),
obj.deg, knotspanS);   %Basis functions
                bt = basisFunTSpl(t, obj.kVertices(i).knotVec(2,:),
obj.deg, knotspanT);   %Basis functions
                dN_s =
derRecur(s,obj.kVertices(i).knotVec(1,:),obj.deg, knotspanS);
%Derivative over x
                dN_t =
derRecur(t,obj.kVertices(i).knotVec(2,:),obj.deg, knotspanT);
%Derivative over y
                BS(i)=bs;
                BT(i)=bt;

                B = sum(bs)*sum(bt); %Blending function
                Nip(i)=B;
                dNip_x(i)=sum(dN_s)*sum(bt); %dB/dx
                dNip_y(i)=sum(bs)*sum(dN_t); %dB/dy

            end
            for i=1:size(obj.kVertices,1)
                if  Nip(i) == 0
                    Bip(i)=0;
                else
                    Bip(i)=Nip(i)/sum(Nip);
                end
                dBip_x(i)=(dNip_x(i)*sum(Nip)-
Nip(i)*sum(dNip_x))/(sum(Nip))^2;
                dBip_y(i)=(dNip_y(i)*sum(Nip)-
Nip(i)*sum(dNip_y))/(sum(Nip))^2;
            end
```

```
        end
```

A.4.3 Συνάρτηση υπολογισμού των ρητών συναρτήσεων T-spline με βάρος

Αποτελεί ανάλογη συνάρτηση υπολογισμού των ρητών συναρτήσεων T-spline με χρήση των συντελεστών βάρους και του τοπικού μητρώου IEN_e κάθε στοιχείου e.

Μεταβλητές εισόδου: `obj,IEN_e, s, t`

Μεταβλητές εξόδου: `BS, BT,Bip,dBip_x,dBip_y`

```
function [BS, BT,Bip,dBip_x,dBip_y] = blendfunNormW(obj,IEN_e, s, t)
            if isempty(IEN_e)
                nel=size(obj.kVertices,1);
                IEN_e=1:nel;
            else
            nel=length(IEN_e);
            end
            Bip=zeros(nel,1);
            dBip_x=zeros(nel,1);
            dBip_y=zeros(nel,1);
            BS=zeros(nel,1);
            BT=zeros(nel,1);
            denom=0; denom_X=0; denom_Y=0;
            for i=1:nel
                knotspanS = 1;
                knotspanT = 1;

                bs = basisFunTSpl(s,
obj.kVertices(IEN_e(i)).knotVec(1,:), obj.deg, knotspanS);   %Basis
functions
                bt = basisFunTSpl(t,
obj.kVertices(IEN_e(i)).knotVec(2,:), obj.deg, knotspanT);   %Basis
functions
                dN_s =
derRecur(s,obj.kVertices(IEN_e(i)).knotVec(1,:),obj.deg, knotspanS);
%Derivative over x
                dN_t =
derRecur(t,obj.kVertices(IEN_e(i)).knotVec(2,:),obj.deg, knotspanT);
%Derivative over y
                BS(i)=bs;
                BT(i)=bt;
                B = sum(bs)*sum(bt); %Blending function
                Nip(i)=B;
                dNip_x(i)=sum(dN_s)*sum(bt); %dB/dx
                dNip_y(i)=sum(bs)*sum(dN_t); %dB/dy
                denom=denom+obj.kVertices(IEN_e(i)).weight*Nip(i);

denom_X=denom_X+obj.kVertices(IEN_e(i)).weight*dNip_x(i);

denom_Y=denom_Y+obj.kVertices(IEN_e(i)).weight*dNip_y(i);
            end
            for i=1:nel
                if  Nip(i) == 0
                    Bip(i)=0;
                else
                    Bip(i)=Nip(i)*obj.kVertices(IEN_e(i)).weight/denom;
%true
```

```
                end
                dBip_x(i)=(dNip_x(i)*denom-
Nip(i)*denom_X)*obj.kVertices(IEN_e(i)).weight/(denom)^2;
                dBip_y(i)=(dNip_y(i)*denom-
Nip(i)*denom_Y)*obj.kVertices(IEN_e(i)).weight/(denom)^2;

            end
        end
```

A.4.4 Συνάρτηση υπολογισμού της επιφάνειας T-spline με βάρος

Αποτελεί την τροποποιημένη εκδοχή της αρχικής συνάρτησης `evaluate` και υπολογίζει την επιφάνεια T-spline λαμβάνοντας υπ' όψιν τους συντελεστές βάρους σύμφωνα με τη σχέση Χ.

Μεταβλητές εισόδου: `obj, s, t`

Μεταβλητές εξόδου: `P`

```
  %Evaluate t-spline
        function P = evaluate(obj, s, t)
            sumPB = 0;
            sumB = 0;
            for i=1:size(obj.kVertices,1)
                %  s

                %check if s,t is in parameter space of vertice
                if s >= obj.kVertices(i).knotVec(1,1) && ...
                        s <= obj.kVertices(i).knotVec(1,end) && ...
                        t >= obj.kVertices(i).knotVec(2,1) && ...
                        t <= obj.kVertices(i).knotVec(2,end)

                    bs = basisFunTSpl(s, obj.kVertices(i).knotVec(1,:),
obj.deg, knotspanS);
                    bt = basisFunTSpl(t, obj.kVertices(i).knotVec(2,:),
obj.deg, knotspanT);
                        B = sum(bs*bt);
                    sumPB = sumPB + obj.kVertices(i).cPoint *
obj.kVertices(i).weight * B;

                    sumB = sumB + obj.kVertices(i).weight* B;
                end
            end
            if sumB == 0
                [P] = sumPB;
            else
                [P] = sumPB/sumB;

            end
        end
```

A.5 Λοιπές συναρτήσεις

A.5.1 Συνάρτηση υπολογισμού του επεκτεταμένου κομβοδιανύσματος

Αποτελεί βοηθητική συνάρτηση, η οποία στην περίπτωση των τοπικών κομβοδιανυσμάτων, προσθέτει τους υπολειμμένους κόμβους στο αρχικό κομβοδιάνυσμα μέχρι οι ακραίες κομβικές τιμές να αποκτήσουν πολλαπλότητα $p + 1$. Επιπλέον, εξάγει το πλήθος των κόμβων, `nt`, οι οποίοι προστέθηκαν στην αρχή του τοπικού κομβοδιανύσματος.

Μεταβλητές εισόδου: `knot,deg`

Μεταβλητές εξόδου: `k,nt`

```
function [k,nt]=knotextend(knot,deg)
    k=knot;
    multiplicity1=nnz(knot(:)==knot(1));
    if multiplicity1<deg+1
        for i=1:deg+1-multiplicity1
        k=[k(1:i) knot(1) k(i+1:end)];
        end
    end

multiplicity2=nnz(knot(:)==knot(end));

    if multiplicity2<deg+1
        for i=1:deg+1-multiplicity2
        k(end+1)= knot(end);
        end
    end
nt=deg+1-multiplicity1;
end
```

A.5.2 Συνάρτηση εξαγωγής Bézier για μονοδιάστατη συνάρτηση B-spline/T-spline

Η συγκεκριμένη συνάρτηση υπολογίζει για δεδομένη μονοδιάστατη συνάρτηση B-spline και δεδομένο στοιχείο τη γραμμή από τον τοπικό τελεστή εξαγωγής Bézier του στοιχείου που αντιστοιχεί στη συγκεκριμένη συνάρτηση. Επιπλέον, εξάγονται το πλήθος των στοιχείων που ορίζονται από την εξαγωγή της συνάρτησης, `nb`, και ο αύξων αριθμός της γραμμής, `pos`, η οποία υπολογίστηκε από το μητρώο του τελεστή εξαγωγής του στοιχείου. Χειρίζεται, επίσης, συναρτήσεις T-spline στη μια διάσταση, καθώς επιτρέπει την εξαγωγή σε διαστήματα κι εκτός των κομβοδιαστημάτων που ορίζονται από το κομβοδιάνυσμα `uknot`. Τα εν λόγω διαστήματα προκύπτουν από την προσθήκη `m` το πλήθος τιμών `U` στις θέσεις που ορίζονται από το διάνυσμα `spans`. Η συνάρτηση βασίζεται στον αλγόριθμο που παρουσιάστηκε στο [13].

Μεταβλητές εισόδου: `p,m,uknot,spans,U,knotspan`

Μεταβλητές εξόδου: `c,nb,pos`

```
function [c,nb,pos]=Bezier_TSpline(p,m,uknot,spans,U,knotspan)
% An algorithm to compute the univariate extraction operator rows
corresponding to
% a single univariate T-spline basis function
%
[Ubar,nt]=knotextend(uknot,p); %extended knot vector

% Initialization variables:
a = p+1; %a=4;
b = a+1; %b=5;
nb = 1; %row/element

c(1,:) = 0;
c(1,p+1) = 0;
c(1,nt+1) = 1;
mbar =   length(Ubar)-p+m;
ki = 1;
si = 1;
while b < mbar

    c(nb+1,:) = 0;
    if nt==p
          if p+1-nb>0
   c(nb+1,p+1-nb)=1;
          end
    end
     if ismember(1,spans)
         c(nb+1,p+1-nb+1)=1;
     end

    % Initialize the next extraction operator row.
    % Count multiplicity of the knot at location b.

    add = 0;

    if si <= m && spans(si) ==ki
        mult = 0;
        add = 1;
```

```
        % Add the new knot to the knot vector.
        Ubar;

        Ubar(b+1:mbar+p-m+si) = Ubar(b:mbar+p-m+si-1);
        Ubar(b) = U(si);
        si = si + 1;

    else
        ki = ki+1;
        i = b;
        while b < mbar && Ubar(b+1) == Ubar(b)
            b = b+1;
        end
            mult = b-i+1;

    end
        mult;
        Ubar(b);
        if mult < p

            numer = Ubar(b)-Ubar(a);
            for j = p:-1:mult+1

            alphas(j-mult) = numer / (Ubar(a+j+add)-Ubar(a));
            end
            r = p-mult;

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

%             c(nb,save)=0;
                s = mult+j;
                for k=p+1:-1:s+1

                    alpha = alphas(k-s);
                    c(nb,k) = alpha*c(nb,k) + (1.0-alpha)*c(nb,k-1);

%                 [c(2,p-1) c(2,p)]
                end
                if b < mbar
                % Update overlapping coefficients of the next operator
row.

                    c(nb+1,save) = c(nb,p+1);

                end
%             disp('----------------------------');
            end
            nb = nb + 1; %Finished with the current operator.
            if b < mbar
                % Update indices for the next operator.
                a = b;
                b = b+1;
            end
        end
```

```
end
local_el=unique(Ubar);
[R,~] = find(bsxfun(@eq,local_el(:),knotspan));
pos=R(1);
c=c(pos,:);
% disp(Ubar);
end
```

A.5.3 Συνάρτηση εξαγωγής Bézier για διδιάστατη συνάρτηση T-spline

Η εν λόγω συνάρτηση αποτελεί διδιάστατο ανάλογο της συνάρτησης εξαγωγής Bézier για μονοδιάστατη συνάρτηση T-spline. Εξάγει, δηλαδή, τη γραμμή του τοπικού τελεστή εξαγωγή Bézier που αντιστοιχεί στη συνάρτηση T-spline καθώς και τις γραμμές των τοπικών τελεστών εξαγωγής ανά κατεύθυνση.

Μεταβλητές εισόδου: `p,q,m1,m2,uknot,vknot,spans1,spans2,U1,U2,elmnts_pts,e`

Μεταβλητές εξόδου: `C,Cxi,Cet`

```
function [C,Cxi,Cet] =
bezierExtraction2D(p,q,m1,m2,uknot,vknot,spans1,spans2,U1,U2,elmnts_pts
,e)
%
% Bézier extraction operators for a 2D T-Spline.

knotspan1=[elmnts_pts(e,1); elmnts_pts(e,3)]'; %element's coordinate x
knotspan2=[elmnts_pts(e,2);elmnts_pts(e,6)]' ; % element's coordinate y

[Cxi,~,~]=Bezier_TSpline(p,m1,uknot,spans1,U1,knotspan1);
[Cet,~,~]=Bezier_TSpline(q,m2,vknot,spans2,U2,knotspan2);


size1 = size(Cxi,2); %row
size2 = size(Cet,2); %column

%---------------
%number of elements in the support
C = zeros(1,size1*size2);
Ctemp=kron(Cet(1,:),Cxi(1,:));
            C(:)=Ctemp;
end
```

A.5.4 Συνάρτηση υπολογισμού τοπικού τελεστή εξαγωγής Bézier για διδιάστατο T-spline

Η παρακάτω συνάρτηση συναρμολογεί τον τοπικό τελεστή εξαγωγής Bézier e, πραγματοποιώντας εξαγωγή Bézier για όλες τις συναρτήσεις που είναι μη μηδενικές στο στοιχείο, σύμφωνα με το μητρώο `IEN_array`.

Μεταβλητές εισόδου: `p,q,nbas,IEN_array,e,elmnts_pts,U_vec,V_vec`

Μεταβλητές εξόδου: `C_e`

```
function C_e=BE_Ce(p,q,nbas,IEN_array,e,elmnts_pts,U_vec,V_vec)

C_e=zeros(nbas,(p+1)*(q+1));
IEN_e=nonzeros(IEN_array(e,:));

%     nel=length(IEN_e);
%     p=3; q=3;

    x=[elmnts_pts(e,1); elmnts_pts(e,3)]'; %element's coordinate x
    y=[elmnts_pts(e,2); elmnts_pts(e,6);]' ; %element's coordinate y

    l=1;
    for i=1:length(IEN_e) %loop over element's non zero shape functions

          uknot=U_vec(IEN_e(i),:); vknot=V_vec(IEN_e(i),:);

        if    any(~ismember(x(1:2),uknot))    ||    any(~ismember([y(1)
y(2)],vknot))
            pos1=find(ismember(x(1:2),uknot)==0);
            pos2=find(ismember([y(1) y(2)],vknot)==0);
            m1=length(pos1); m2=length(pos2);
            U1=x(pos1); %knot values to be inserted into uknot
            U2=y(pos2); %knot values to be inserted into vknot

            spans1=zeros(1,m1);
            spans2=zeros(1,m2);
            u_un=unique(uknot);
            v_un=unique(vknot);
            for w=1:length(U1)
                for m=1:length(u_un)
                    if u_un(m)<U1(w)
                        spans1(w)=spans1(w)+1;
                    else
                        break
                    end
                end
            end
```

```
            for w=1:length(U2)
                for m=1:length(v_un)
                    if v_un(m)<U2(w)
                        spans2(w)=spans2(w)+1;
                    else
                        break
                    end
                end
            end
    else
            m1=0; m2=0; spans1=[]; spans2=[]; U1=[]; U2=[];
        end
    [Ctemp,Cxi,Cet]                                                    =
bezierExtraction2D(p,q,m1,m2,uknot,vknot,spans1,spans2,U1,U2,elmnts_pts
,e);

    C_e(l,:)=Ctemp(:);
    l=l+1;

    end
end
```

A.5.5 Συνάρτηση υπολογισμού συναρτήσεων T-spline με εφαρμογή εξαγωγής Bézier

Η συνάρτηση υπολογίζει τις συναρτήσεις βάσης T-spline και τις μερικές παραγώγους τους, για τις μη μηδενικές συναρτήσεις του δεδομένου στοιχείου, αξιοποιώντας τον προ-υπολογισμένο τελεστή εξαγωγής Bézier. Αποτελεί παραλλαγή της συνάρτησης υπολογισμού των συναρτήσεων NURBS που παρουσιάστηκε στο [6].

Μεταβλητές εισόδου: `xi,eta,poly,IEN_array,e,w,C_e`

Μεταβλητές εξόδου: `Rb,dR`

```
% This function forms basis functions and derivatives
%
% Output: Rb - TSpline basis functions
% dR - derivatives of TSpline basis functions
%
% Input: xi - coord. of 1st parametric direction
% eta - coord. of 2nd parametric direction
% poly - polynomial order
% e - current element number
% IEN_array - element topology of current element
% w – T-spline weights
% C_e - Bézier extraction operators
function [Rb,dR]=TSplineBasis(xi,eta,poly,IEN_array,e,w,C_e)
      % Bernstein basis functions and derivatives
      IEN_e=nonzeros(IEN_array(e,:));

      [Bbxi,dBxi]=BernsteinBasis(xi,poly);
      [Bbeta,dBeta]=BernsteinBasis(eta,poly);
      [Bb,dB]=BernsteinBasisBivariate(Bbxi,Bbeta,dBxi,dBeta,poly);
      % Weight function and derivatives of weight function, Bézier
      element
      wb=C_e(1:length(IEN_e),:)'*w(IEN_e); % Bézier weights
      Wb=Bb*wb;
      dWbxi=dB(1,:)*wb;
      dWbeta=dB(2,:)*wb;
      % T-spline basis functions and derivatives
      Rb=diag(w(IEN_e))*C_e(1:length(IEN_e),:)*Bb'/Wb;
      Rb=Rb';
      dR(:,1)=diag(w(IEN_e))*C_e(1:length(IEN_e),:)*(dB(1,:)'/Wb-
      dWbxi*Bb'/Wb^2);
      dR(:,2)=diag(w(IEN_e))*C_e(1:length(IEN_e),:)*(dB(2,:)'/Wb-
      dWbeta*Bb'/Wb^2);
      dR=dR';
end
```

A.5.6 Συνάρτηση συναρμολόγησης μητρώου στιβαρότητας με εφαρμογή εξαγωγής Bézier

Η συνάρτηση συναρμολογεί το καθολικό μητρώο στιβαρότητας, εφαρμόζοντας αξιοποιώντας τον τοπικό τελεστή εξαγωγής κάθε στοιχείου Bézier του πλέγματος T-spline. Επιπλέον, η περιοχή της επιφάνειας T-spline, `area`, εκτυπώνεται για τον εντοπισμό σφαλμάτων. Αποτελεί παραλλαγή της συνάρτησης συναρμολόγησης του μητρώου στιβαρότητας με συναρτήσεις NURBS που παρουσιάστηκε στο [6].

```
% This function forms global stiffness matrix
%
% Output: K - global stiffness matrix
%
% Input: IEN_array - element topology: numbering of control points
% P - coordinates of TSpline control points
% K - empty global stiffness matrix
% nel - total number of elements
% ncp - number of control points
% poly - polynomial order
% w - weights of T-spline control points
% C - Bézier extraction operators
function [K,area]=FormK_H(IEN_array,P,K,nel,poly,w,C)
area=0;
    [G,W]=Gauss(poly); % Call Gauss points and weights
    for e=1:nel
        IEN_e=nonzeros(IEN_array(e,:))'; % Element topology of current
el.
        nbas=length(IEN_e);
        C_e=C(:,:,e);
        eDof=[IEN_e]; % eDof: first x, then y
        k=0;
        for g=1:size(G,1) % For each Gauss point
            xi=G(g,1); % Gauss coord. reference element
            eta=G(g,2); % Gauss coord. reference element
            [~,dR]=TSplineBasis(xi,eta,poly,IEN_array,e,w,C_e);
            [J,dxy]=Jacobian(dR,P,IEN_e);
            B=zeros(2,length(IEN_e)); % B matrix
            B(1,1:length(IEN_e))=dxy(1,:);
            B(2,1:length(IEN_e))=dxy(2,:);
            k=k+B'*B*det(J)*W(g);

            area = area + det(J)*W(g);
        end

        K(eDof,eDof)=K(eDof,eDof)+k;
     end
end
```