ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Κατασκευή λογισμικού για εκτέλεση επιστημονικών υπολογισμών με την χρήση εργαλείων CAD

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΑΘΑΝΑΣΙΑΔΗ ΠΑΝΑΓΙΩΤΗ
Α.Ε.Μ 4003

Επιβλέπων

ΙΩΑΝΝΗΣ ΤΟΥΛΟΠΟΥΛΟΣ

Εγκρίθηκε από την τριμελή επιτροπή

ΙΩΑΝΝΗΣ ΤΟΥΛΟΠΟΥΛΟΣ, ΜΙΧΑΗΛ ΣΤΑΜΠΟΥΛΤΖΗΣ, ΒΑΓΙΩΝΑΣ ΧΡΗΣΤΟΣ

Καστοριά

22 Ιανουαρίου 2025

Ευχαριστίες

Η παρούσα εργασία αποτελεί πτυχιακή διατριβή στο πλαίσιο του προπτυχιακού προγράμματος του Τμήματος Πληροφορικής του Πανεπιστημίου Δυτικής Μακεδονίας.

Θα ήθελα να ευχαριστήσω, πρώτα απ' όλα, τον επιβλέποντα καθηγητή της πτυχιακής μου εργασίας, κ.Τουλόπουλο Ιωάννη, για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα ιδιαίτερα ενδιαφέρον θέμα, καθώς και για την έμπνευση που μου προσέφερε να εμβαθύνω στον τομέα των μαθηματικών. Μέσα από την πολύτιμη καθοδήγησή του και τον τρόπο εργασίας του, απέκτησα μια καλή ιδέα για την ερευνητική διαδικασία.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, διότι χωρίς τη στήριξή τους, δεν θα είχα τον χρόνο να ολοκληρώσω την παρούσα εργασία.

Περίληψη

Στην παρούσα διπλωματική εργασία παρουσιάζεται η υλοποίηση αριθμητικών μεθόδων σε C/C++ για την προσέγγιση λύσεων διαφορικών εξισώσεων με συνοριακές συνθήκες. Αρχικά, εισάγουμε την έννοια των B-splines ως εργαλεία προσέγγισης, χάρη στις γεωμετρικές τους ιδιότητες. Επεξηγούνται θεμελιώδεις έννοιες για τις B-splines καμπύλες , όπως η συναρτήσεις βάσης,το διάνυσμα κόμβων, η τάξη της καμπύλης, τα σημεία ελέγχου και το κυρτό περίβλημα. Στη συνέχεια, κατασκευάζουμε ένα σύστημα $Ax = b$, το οποίο επιλύουμε μέσω της μεθόδου Cholesky.

Για λόγους απλότητας, όλα τα παραπάνω υλοποιούνται σε μονοδιάστατο χώρο.Οι καμπύλες και οι συναρτήσεις, καθώς και οι κατασκευαστές των σχημάτων στα επόμενα κεφάλαια, είναι όλα γραμμένα στη γλώσσα προγραμματισμού C/C++, ενώ γίνεται χρήση των πακέτων Gnu Scientific Library (GSL). Τα γραφικά δημιουργήθηκαν με τη χρήση του προγράμματος Gnuplot.Επιπλέον, τα κομμάτια κώδικα που παρατίθενται είναι διαθέσιμα ως πλήρη προγράμματα με παραδείγματα στην πλατφόρμα GitHub.[1]

Λέξεις Κλειδιά - τάξη της καμπύλης,βαθμός πολυωνύμου,B-spline,κόμβος,Cholesky

[1] https://github.com/PanagiotisAthanasiadis/b-spline-tools-/tree/main/thesis_examples

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

Πρόλογος

Οι διαφορικές εξισώσεις αποτελούν έναν θεμελιώδη τρόπο για την περιγραφή διαφόρων φυσικών φαινομένων γύρω μας. Ωστόσο, στην πράξη, πολλές φορές είναι αδύνατο ή υπολογιστικά πολύ ακριβό να βρούμε λύσεις για αυτές τις εξισώσεις με αναλυτικές μεθόδους. Για τον λόγο αυτό, καταφεύγουμε σε αριθμητικές μεθόδους που προσεγγίζουν τη λύση μέσα σε ένα επιθυμητό όριο σφάλματος.

Με την πάροδο των χρόνων, έχουν αναπτυχθεί διάφορες μέθοδοι για την επίτευξη προσεγγιστικών λύσεων, όπως η μέθοδος των πεπερασμένων στοιχείων, η μέθοδος των πεπερασμένων διαφορών, η μέθοδος gradient discretization και άλλες. Στην παρούσα εργασία, θα ασχοληθούμε με τη μέθοδο των πεπερασμένων στοιχείων και, συγκεκριμένα, με τη χρήση των B-splines.

Ο Επιστημονικός Υπολογισμός είναι το αντικείμενο που λαμβάνει ένα μαθηματικό μοντέλο, δημιουργεί ένα αριθμητικό μοντέλο, προτείνει αλγορίθμους για τον προσδιορισμό της υπολογιστικής λύσης και στη συνέχεια επαληθεύει πόσο σωστά και ακριβή είναι τα αποτελέσματα του υπολογιστή συγκρίνοντάς τα με την πραγματική λύση.

Τα B-splines είναι τμηματικά ορισμένα πολυώνυμα, τα οποία, όταν συνδυάζονται, δημιουργούν μια συνεχή καμπύλη. Σε σύγκριση με άλλα πολυώνυμα, όπως αυτά του Legendre και του Hermite, που συνήθως χρησιμοποιούνται στη μέθοδο αυτή, τα B-splines είναι πολύ πιο εύκολα στον υπολογισμό. Οι καμπύλες που παράγονται με B-splines προσφέρουν υψηλή ομαλότητα και υψηλή συνέχεια στα διάφορα σημεία σύνδεσής τους. Τα B-splines χρησιμοποιούνται για την περιγραφή επιφανειών στα γραφικά υπολογιστών και, ειδικότερα, ως εργαλείο σε διάφορα πακέτα CAD. Σκοπός της εργασίας είναι να χρησιμοποιήσει CAD εργαλεία (δηλαδή B-splines) για την εύρεση προσεγγιστικής λύσης ενός μαθηματικού μοντέλου - προβλήματος διάχυσης.

- Στόχος στην εργασία αυτή είναι να παρουσιάσουμε τους χώρους B-spline

που χρησιμοποιούνται στα CAD για να λύσουμε αριθμητικά προβλήματα διαφορικών εξισώσεων.

- Να δείξουμε πώς παράγεται το διακριτό σχήμα που μας δίνει την προσεγγιστική λύση.
- Να υλοποιήσουμε τη μέθοδο Galerkin με γενικές B-Splines.
- Να δώσουμε λεπτομέρειες για την υλοποίηση του αλγορίθμου σε προγραμματιστικό περιβάλλον C++ και να κατασκευάσουμε το λογισμικό.
- Να εκτελέσουμε αριθμητικά παραδείγματα και να μελετήσουμε την συμπεριφορά των αποτελεσμάτων που δίνει ο υπολογιστής.

Παρακάτω παρουσιάζουμε ένα διάγραμμα ανάπτυξης της εργασίας για την διακριτοποίηση ενός μαθηματικού μοντέλου(Διαφορική-Εξίσωση).

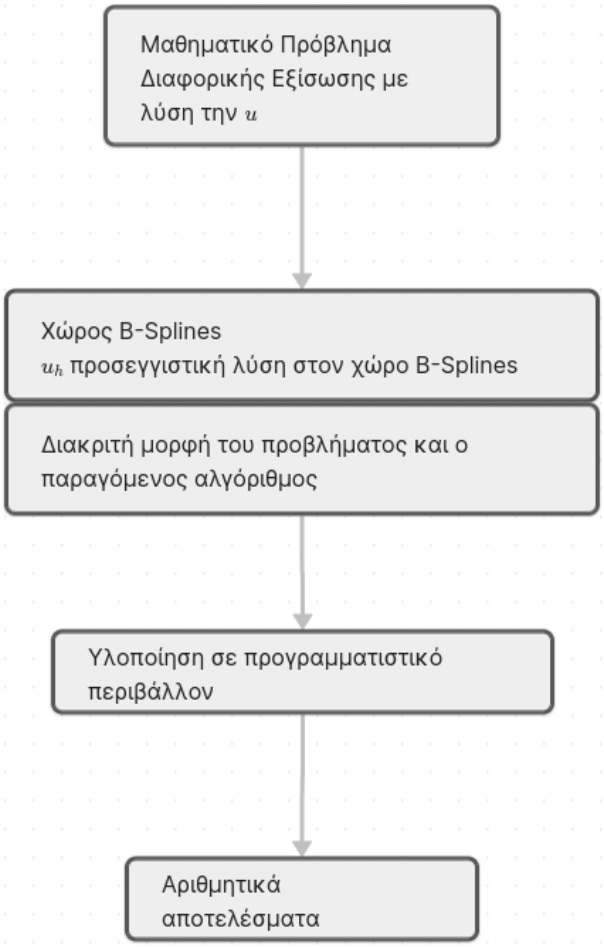

Figure 1: Διάγραμμα ανάπτυξης της εργασίας

Για τους σκοπούς αυτής της εργασίας όλοι οι συσχετιζόμενοι αλγόριθμοι που προκύπτουν στο διάγραμμα 1,έχουν υλοποιηθεί σε προγραμματιστικό περιβάλλον C++ (και σε C χρησιμοποιώντας την βιβλιοθήκη Gsl) για να προκύψει το λογισμικό υπολογισμού της αριθμητικής λύσης ,το οποίο υπάρχει στο Github. Η πλατφόρμα Github προσφέρει την δυνατότητα στους κατασκευαστές λογισμικού να αποθηκεύουν το κωδικά του λογισμικού τους διαδικτυακά με το όφελος της εύκολης διαχείρισης εκδόσεων(Version Control), άμεσης πρόσβασης, αυτομάτου ελέγχου σφαλμάτων(pipelines) και για καλύτερο συντονισμό μεγάλων ομάδων προγραμματιστών.

Κεφάλαιο 1

B-splines

1.1 Ορισμοί: Διάνυσμα κόμβων και B-splines

1.1.1 Διάνυσμα κόμβων

Σύμφωνα με τις απαιτήσεις μας, χωρίζουμε το αρχικό διάστημα $[a, b]$ σε μικρά υποδιαστήματα. Για να το επιτύχουμε αυτό, χρησιμοποιούμε μια μη φθίνουσα ακολουθία πραγματικών αριθμών,τα t_i για $i = 1, ..., p$ όπου $a \leq t_i \leq b$, ονομάζονται σημεία καμπής (breakpoints) ή απλώς κόμβοι. Σε κάθε συνάρτηση βάσης B-spline, που θα οριστεί πάνω στα t_i αντιστοιχεί ένας συγκεκριμένος αριθμός υποδιαστημάτων ο οποίος είναι $d + 1$. Έτσι το διάνυσμα των κόμβων έχει μορφή,

$$T = [a, t_1, t_2, ..., t_{p-1}, b] \quad t_i \leq t_{i+1} \tag{1.1.1}$$

- T διάνυσμα κόμβων
- t_i κόμβοι
- p πλήθος κόμβων δίνεται από την σχέση $p = n + d + 1$, οπού n είναι το πλήθος των σημείων ελέγχου και των συναρτήσεων βάσης των B-splines και d είναι ο βαθμός των συναρτήσεων βάσεις B-splines.

Το πλήθος των συναρτήσεων βάσης n εξαρτάται από τη σχέση $n = p - d + 1$. Στο σχήμα 1.1 και στο σχήμα 1.2 δίνεται ένα παράδειγμα παραμετρικής καμπύλης B-spline. Υπάρχουν διάφορα είδη διανυσμάτων κόμβων,ο τρόπος που κατηγοριοποιούνται εξαρτάται από το μέγεθος των διαστημάτων ανάμεσα στους κόμβους και η πολλαπλότητα των κόμβων m. Αν το μέγεθος όλων των διαστημάτων είναι ίσο, τότε έχουμε ένα ομοιόμορφο διάνυσμα κόμβων αν όχι, τότε έχουμε ένα ανομοιόμορφο. Για παράδειγμα,

$$T_1 = [0, 1, 2, 3, 3, 4] \quad \text{Ομοιόμορφο}$$
$$T_2 = [0, 1, 2, 2, 2, 3.5, 4] \quad \text{Ανομοιόμορφο}$$

Για το διάνυσμα κόμβων T_1 αν επιλέξουμε B-spline βαθμού $d = 2$ είναι $p = 6$ το διάνυσμα της πολλαπλότητας των κόμβων είναι $[1, 1, 1, 2, 1]$

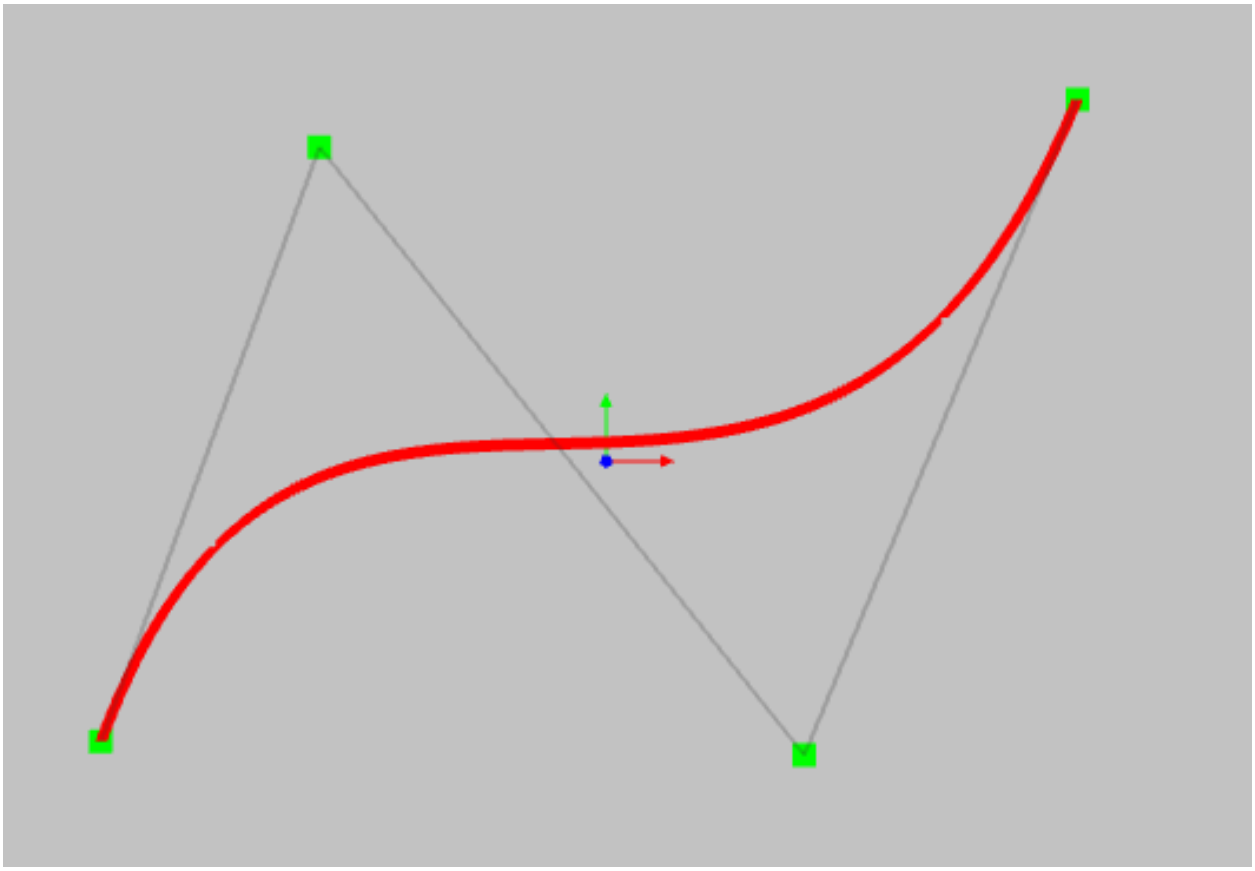

Figure 1.1: B-spline τάξης($k = 3$) με διάνυσμα κόμβων $T = [0, 0, 0, 0, 1, 1, 1, 1]$.
Τα πράσινα τετράγωνα αναπαριστούν τα σημεία ελέγχου με συντεταγμένες
(-7.4,-4.1),(-4.2,4.6),(2.9,-4.3),(6.9,5.3)[1]

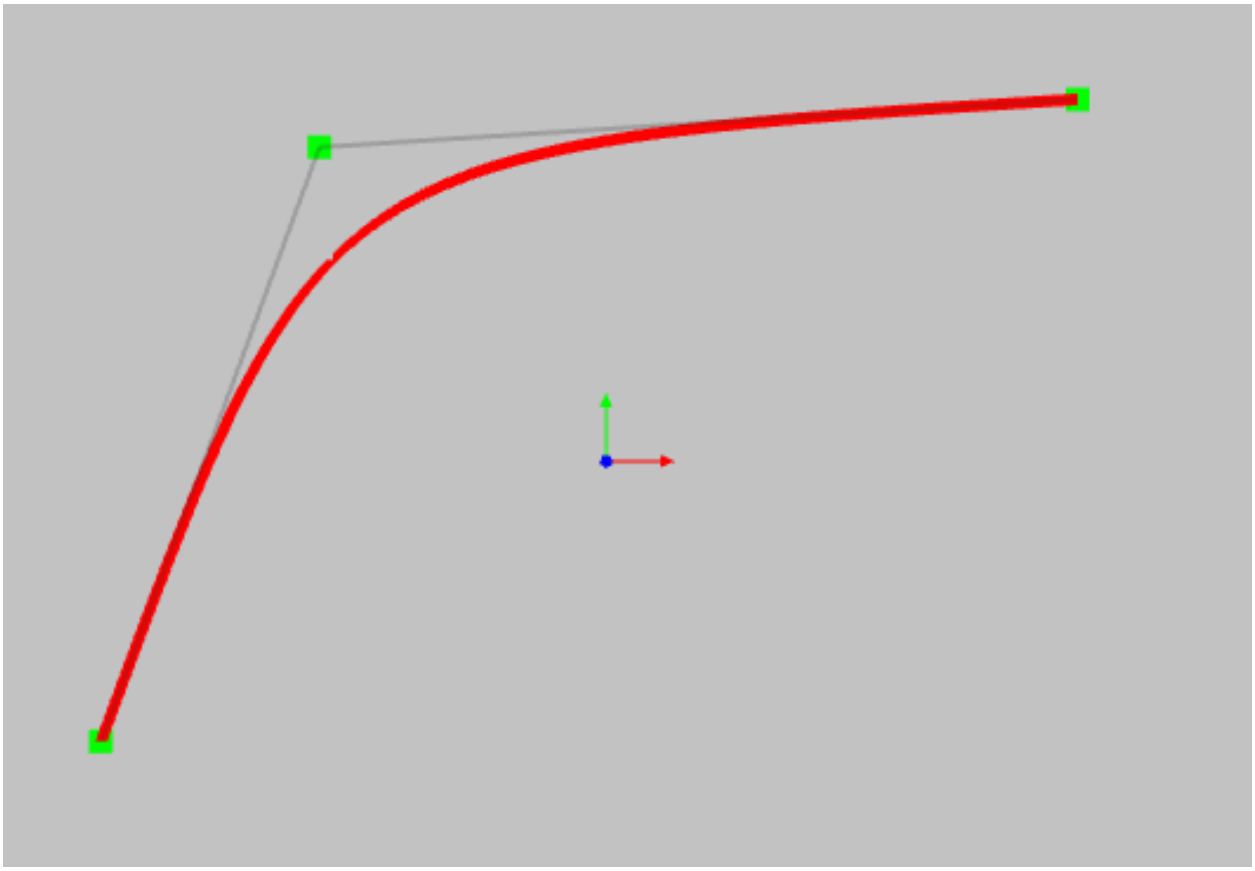

Figure 1.2: B-spline τάξης($k = 3$) με διάνυσμα κόμβων $T = [0, 0, 0, 0, 1, 1, 1, 1]$.
Τα πράσινα τετράγωνα αναπαριστούν τα σημεία ελέγχου με συντεταγμένες
(-7.4,-4.1),(-4.2,4.6),(-4.2,4.6),(6.9,5.3).Η πολλαπλότητα των σημείων ελέγχου είναι 2.[1]

1.1.2 Συναρτήσεις βάσης B-spline

Οι συναρτήσεις βάσης B-spline ορίζονται σε ένα διάστημα $[a, b]$ με την βοήθεια των κόμβων t_i που υπάρχουν στο T. Τα B-splines ή Basis splines είναι τμηματικά ορισμένα πολυώνυμα βαθμού d που ενώνονται στα σημεία των κόμβων με ομαλότητα το πολύ ίση με C^{d-m}. Μια γεωμετρική καμπύλη που συναντάμε στην σχεδίαση γραφικών με την βοήθεια του υπολογιστή (CAD), συνήθως μπορεί να παραμετροποιήθει με την βοήθεια των συναρτήσεων βάσης B-splines, όπως για παράδειγμα η καμπύλη F παρακάτω,

$$F(x) = \sum_{i=0}^{n} c_i B_{i,d}(x) \tag{1.1.2}$$

όπου στην (1.1.2) είναι

- c_i είναι τα σημεία ελέγχου ή συντελεστές των $B_{i,d}$.
- n είναι το πλήθος των σημείων ελέγχου και των συναρτήσεων βάσης των B-splines.
- $B_{i,d}$ είναι συναρτήσεις βάσης B-splines (κατά τμήματα πολυώνυμα) βαθμού d.
- $k = d + 1$ είναι η τάξη της καμπύλης.

Η γεωμετρία, η τάξη της καμπύλης, καθώς και ο αριθμός των σημείων ελέγχου καθορίζονται από το διάνυσμα των κόμβων. Όπως προαναφέραμε η πολλαπλότητα ονομάζεται η επανάληψη κόμβων και την συμβολίζουμε με m, επηρεάζει την ομαλότητα C^{d-m} της καμπύλης στα σημεία των κόμβων όπου ενώνεται, καθώς και τη γεωμετρία της.

Οι συναρτήσεις βάσης είναι τμηματικά ορισμένα πολυώνυμα βαθμού d, και μπορούν να υπολογιστούν από τον παρακάτω αναδρομικό τύπο [2] :

$$B_{i,0}(x) = \begin{cases} 1 & \text{αν } t_i \leq x < t_{i+1}, \\ 0 & \text{αλλιώς,} \end{cases}$$

$$B_{i,d}(x) = \frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x) + \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x) \tag{1.1.3}$$

Στον παραπάνω τύπο θεωρούμε ότι $\frac{0}{0} = 0$.
Το πλήθος των συναρτήσεων βάσης n δίνεται από τη σχέση $n = p - d + 1$

1.1.3 Ιδιότητες του χώρου των συναρτήσεων βάσης B-spline

Οι κυριότερες ιδιότητες των συναρτήσεων βάσης B-spline αναφέρονται στην παρακάτω λίστα.

- Το άθροισμα των συναρτήσεων βάσης είναι πάντα ίσο με 1.

$$\sum_{i=0}^{n} B_{i,d}(x) = 1$$

- Οι τιμές των συναρτήσεων βάσης είναι θετικές ή ίσες με 0.

- Κάθε συνάρτηση έχει συγκεκριμένο φορέα υποστήριξης (δηλαδή είναι διαφορετικές του 0) σε ορισμένα υποδιαστήματα μεταξύ των κόμβων. Αναλυτικά η $B_{i,d}(x)$ έχει φορέα υποστήριξης στο διάστημα $I = [t_i, ..., t_{i+d+1}]$ κόμβους.

- Αν το x βρίσκεται μέσα στο διάστημα $[t_\mu, t_{\mu+1}]$ και $i < \mu - d$ ή $i > \mu$ τότε $B_i, d(x) = 0$

- Αν ο κόμβος t_i επαναλαμβάνεται m φορές $(m \leq d)$ η κάθε $B_{i,d}(x)$, έχει ομαλότητα C^{d-m}. Αυτό ονομάζεται ομαλότητα (Smoothness) της $B_{i,d}$ στον κόμβο t_i .

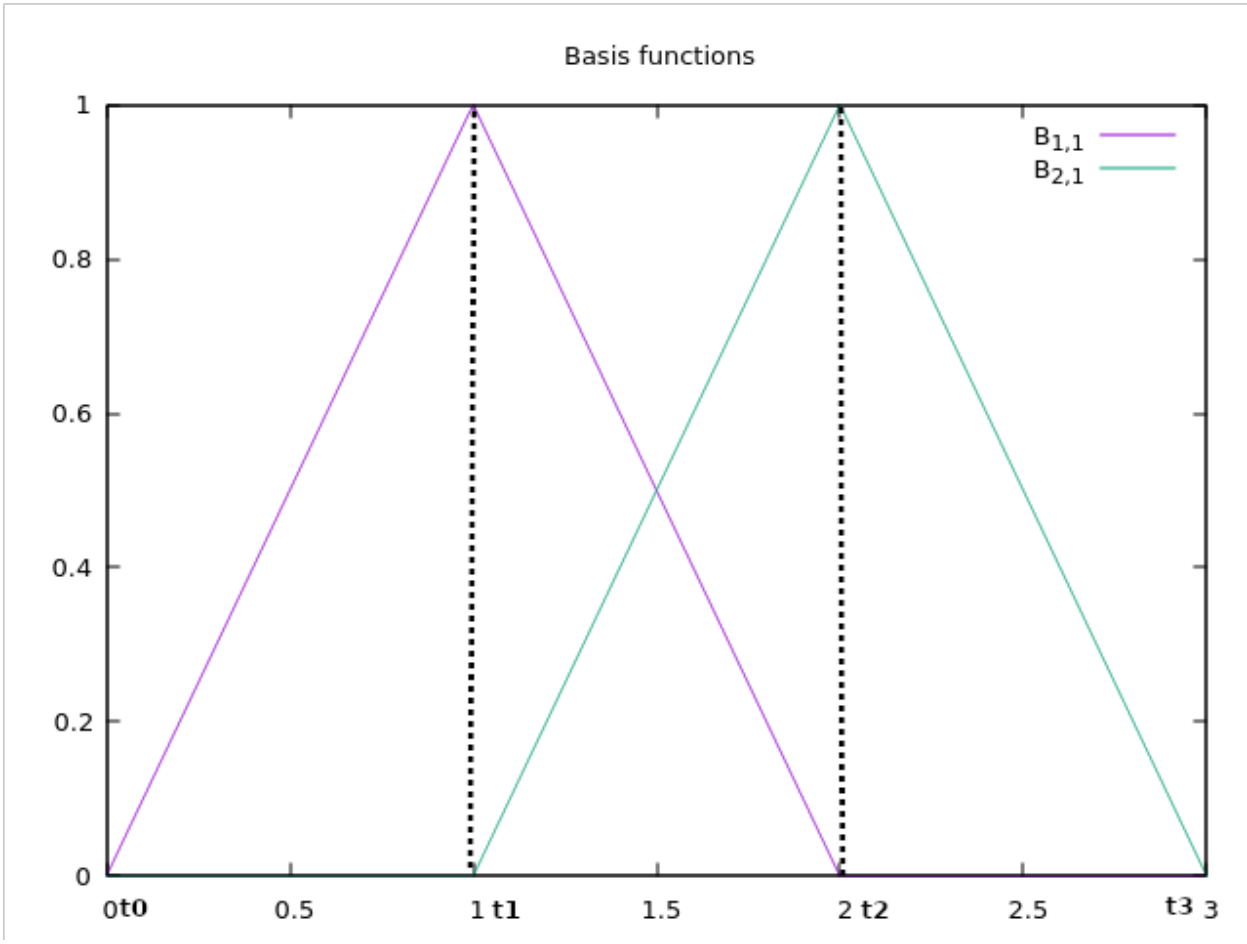

Figure 1.3: Γραμμικές συναρτήσεις βάσης($d = 1$) με διάνυσμα κόμβων $T = [0, 1, 2, 3]$. Οι κατακόρυφες διακεκομμένες γραμμές οριοθετούν την επιρροή κάθε συνάρτησης.

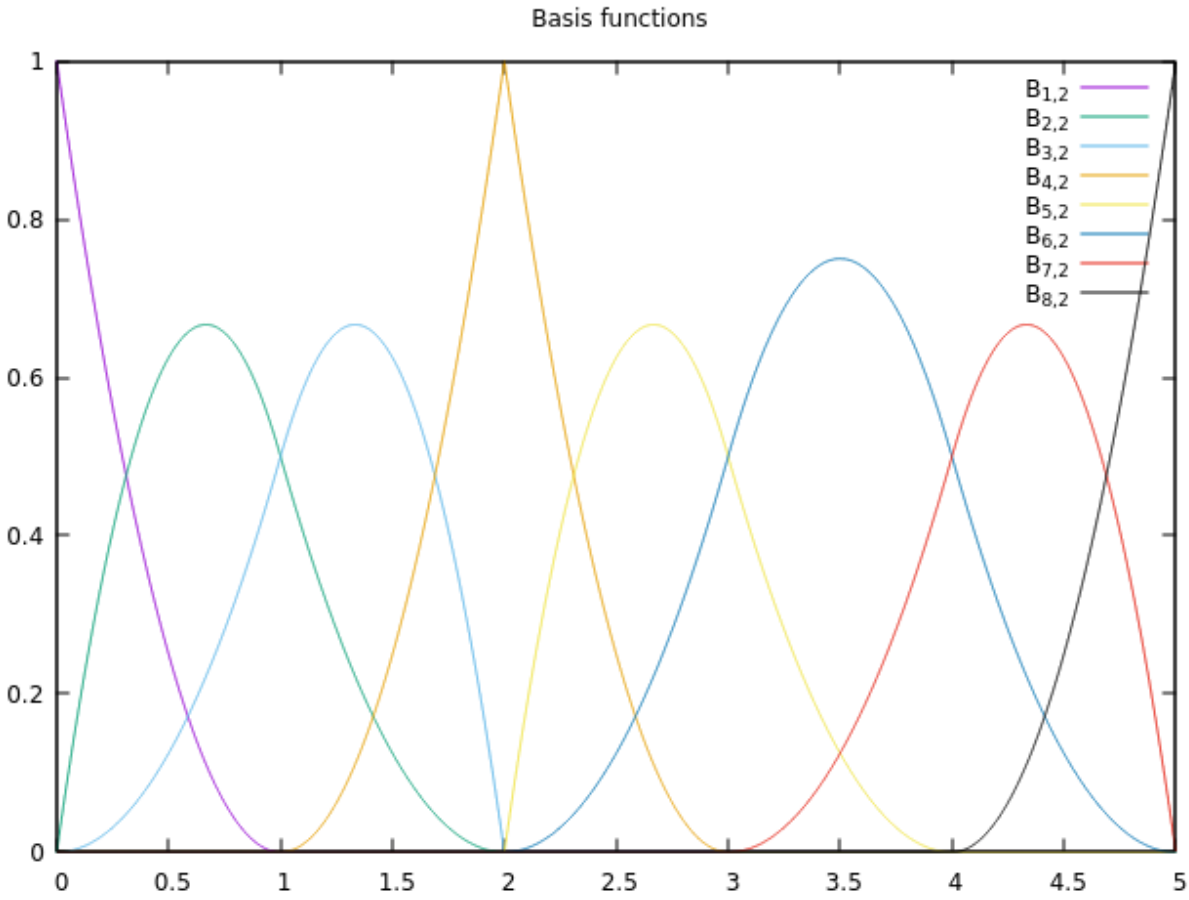

Figure 1.4: Δευτέρου βαθμού συναρτήσεις βάσης($d = 2$) με διάνυσμα κόμβων $T = [0, 0, 0, 1, 2, 2, 3, 4, 5, 5, 5]$.

Ο χώρος των συναρτήσεων βάσης ορίζεται ως,

$$V = Sp(T) = span\{B_{0,d}, B_{1,d}, ..., B_{n,d}\} \tag{1.1.4}$$

Το επόμενο κομμάτι κώδικα υλοποποιεί τις σχέσεις (1.1.3).Τα παρακάτω κομμάτια κώδικα C++ είναι γραμμένα αυτοτελώς και παρατίθενται στο GitHub[1] μου, σαν πλήρη προγράμματα που μπορούν να μεταγλωτιστούν και να εκτελεστούν.

Υλοποίηση του (1.1.3) σε C++

```
double bsp(int i, int ord, double x, vector<double> kns) {
  // Check for illegal value of i
  if (i < 0 || i > kns.size() - ord - 1) {
    cout << "illegal i value: i=" << i << "; kns.size()-ord=" << kns.size() << "-" << ord << "="
         << kns.size() - ord << endl;
    return numeric_limits<double>::quiet_NaN();
  }
  // Return 0 if x is outside the interval defined by the knots
  if (x < kns[i] || x > kns[i + ord])
    return 0.0;
  // Remove repeated knots
  int k = kns.size() - 1;
  while (kns[k] == kns[k - 1])
    k--;
  k--;
  // If ord is 1, return 1 if x is within the interval defined by the knots
  if (ord == 1) {
    if (i != k) {
      return ((kns[i] <= x && x < kns[i + 1]) ? 1.0 : 0.0);
    } else {
      if (i == k) {
        return ((kns[i] <= x && x <= kns[i + 1]) ? 1.0 : 0.0);
      } else
        return numeric_limits<double>::quiet_NaN();
      ;
    }
  }
  // If ord is greater than 1, return the sum of two recursive calls
  else {
    return (gdiv((x - kns[i]) * bsp(i, ord - 1, x, kns),
                 kns[i + ord - 1] - kns[i]) +
            gdiv((kns[i + ord] - x) * bsp(i + 1, ord - 1, x, kns),
                 kns[i + ord] - kns[i + 1]));
  }
}
```

[1] https://github.com/PanagiotisAthanasiadis/b-spline-tools-/tree/main/thesis_examples

1.2 Σημεία Ελέγχου

Όπως αναφέραμε, κάθε συνάρτηση έχει φορέα υποστήριξης σε ορισμένα υποδιαστήματα κόμβων. Έτσι, ανάλογα, και τα σημεία ελέγχου έχουν φορέα υποστήριξης στη συνάρτηση που είναι ορισμένη στο συγκεκριμένο υποδιάστημα:

$$[t_i, t_{i+d+1}] \tag{1.2.1}$$

- t_i κόμβοι από το διάνυσμα κόμβων T

Από τον τύπο (1.1.2) συμπεραίνουμε ότι το πλήθος των σημείων ελέγχου είναι ίσο με το πλήθος των συναρτήσεων βάσης. Οι τιμές τους καθορίζονται ανάλογα με την επιθυμητή γεωμετρία των καμπυλών π.χ δες τα σχήματα 1.1 και 1.2.

1.2.1 Ιδιότητες

- Δεν υπάρχει περιορισμός στις τιμές των σημείων ελέγχου, οπότε μπορεί να προκύψει δύο ή περισσότερα σημεία να έχουν την ίδια τιμή. Αυτό καθορίζει το σχήμα της καμπύλης.

1.3 Παραγώγιση

Η παράγωγος μιας καμπύλης B-spline με συναρτήσεις βάσης βαθμού d είναι μια καμπύλη B-spline, όπου έχει συναρτήσεις βάσης βαθμού $d-1$ και μοιράζεται το ίδιο διάνυσμα κομβών. Ο παρακάτω τύπος βασίζεται στον τύπο (1.1.3) και υπολογίζει την παράγωγο των συναρτήσρων βάσης.

$$B'_{i,d}(x) = \frac{d}{t_{i+d} - t_i} B_{i,d-1}(x) - \frac{d}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x) \tag{1.3.1}$$

Αν τα σημεία ελέγχου είναι c_i, τότε τα νέα σημεία ελέγχου παράγονται ως εξής:

$$Q_i = \frac{d}{t_{i+d+1} - t_{i+1}} (c_{i+1} - c_i) \tag{1.3.2}$$

Οπότε, ο τύπος της παραγώγου της καμπύλης B-spline είναι:

$$F'(x) = \sum_{i=0}^{n-1} B_{i,d-1} Q_i \tag{1.3.3}$$

Υλοποίηση του (1.3.1) σε C++

```
double bsp_deriv(int i, int ord, double x, int nk, vector<double> kns)
{
  // Check for illegal value of i
  if(i<0 || i>nk-ord-1){
    cout<<"illegal i value: i="<<i<<"; nk-ord="<<nk<<"-"<<ord<<"="<<nk-ord<<endl;
    return numeric_limits<double>::quiet_NaN();
  }
  // Return 0 if x is outside the interval defined by the knots
  if(x<kns[i] || x>kns[i+ord]) return 0.0;
  // Remove repeated knots
  int k=nk-1;
  while(kns[k]==kns[k-1]) k--;
  k--;
  // If ord is 1, return 1 if x is within the interval defined by the knots
  if(ord == 1){
    if(i != k){
      return((kns[i]<=x && x<kns[i+1]) ? 1.0 : 0.0);
    }else{
      if(i == k){
        return((kns[i]<=x && x<=kns[i+1]) ? 1.0 : 0.0);
      }else return numeric_limits<double>::quiet_NaN();;
    }
  }
  // If ord is greater than 1, return the difference of two recursive calls
  else{
    return (gdiv(ord,kns[i+ord]-kns[i]) * bsp(i, ord-1, x, nk, kns))-
    (gdiv(ord,kns[i+ord+1]-kns[i+1]) * bsp(i+1, ord-1, x, nk, kns));
  }
}
```

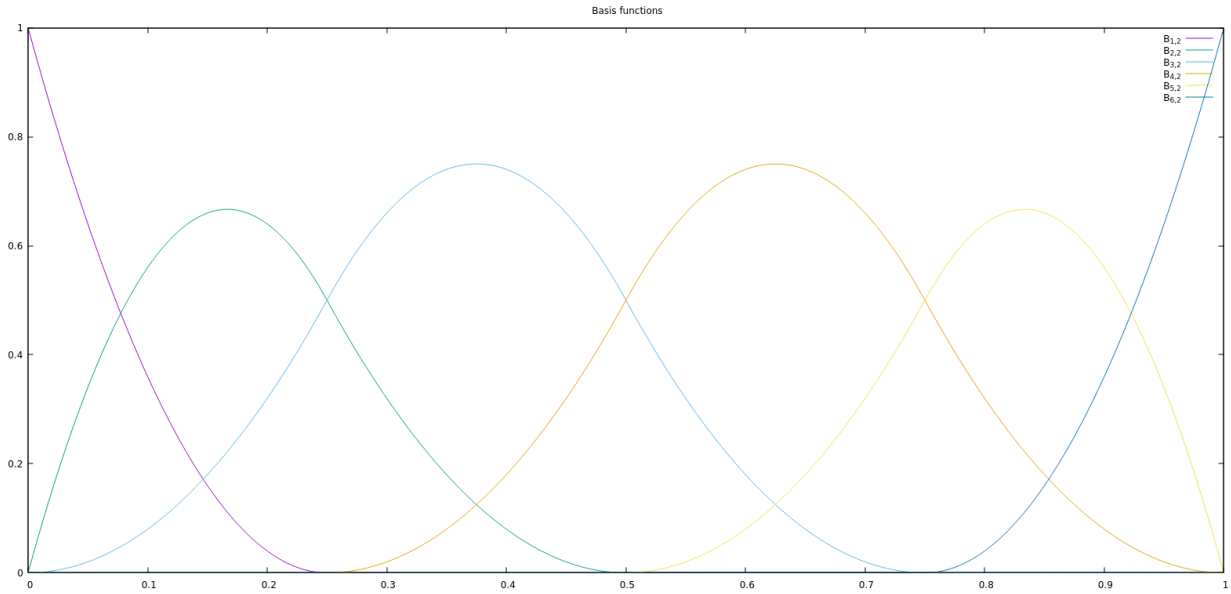

Figure 1.5: B-spline συναρτήσεις βάσης βαθμού $d = 2$ με διάνυσμα κόμβων $T = [0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1]$.

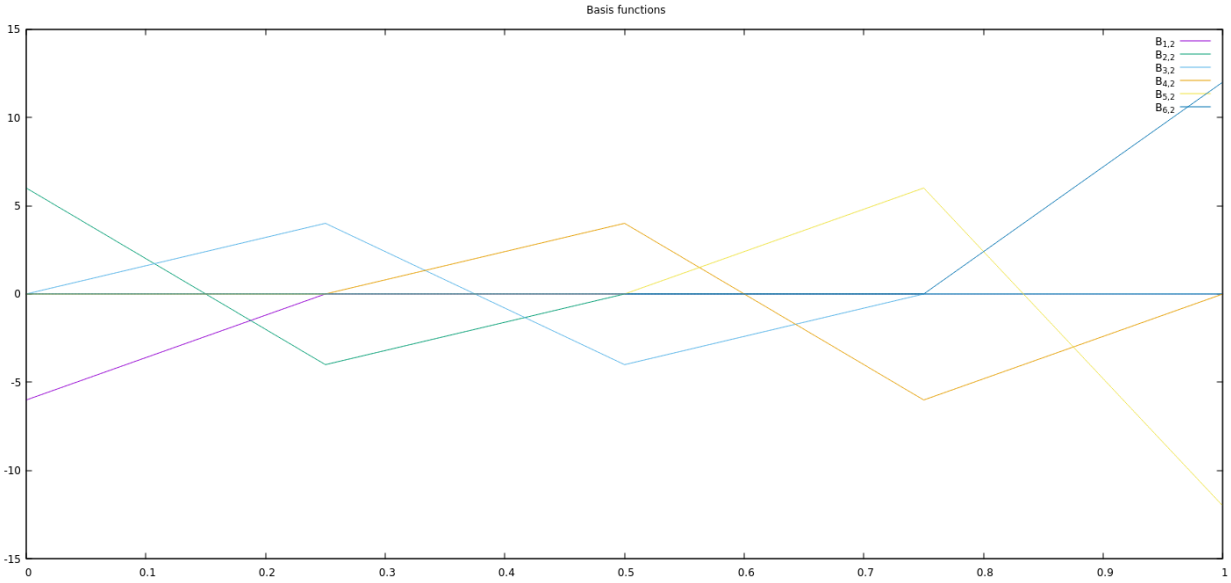

Figure 1.6: B-spline συναρτήσεις βάσης βαθμού $d = 1$ με διάνυσμα κόμβων $T = [0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1]$.

1.4 Αριθμητική Ολοκλήρωση

Ένα ακόμη απαραίτητο εργαλείο είναι η ολοκλήρωση και συγκεκριμένα, η αριθμητική ολοκλήρωση, για παράδειγμα για την καμπύλη που ορίσαμε στην (1.1.2) έχουμε,

$$\int_{t_i}^{t_{i+1}} F(x)dx = \sum_{j=0}^{n} c_j \int_{t_i}^{t_{i+1}} B_{j,d}(x)dx \tag{1.4.1}$$

Η αριθμητική ολοκλήρωση, συγκεκριμένα με τη μέθοδο Gauss-Legendre είναι,[3]

$$\tilde{Q}_\nu(F) = \frac{t_{i+1}-t_i}{2} \sum_{a=1}^{\nu} w_a F(\frac{t_{i+1}-t_i}{2} x_a + \frac{t_{i+1}+t_i}{2}) \tag{1.4.2}$$

Όπου στην (1.4.2)

- w_j βάρη
- x_j ρίζες του ν-ουστου πολυωνύμου Legendre στο $[-1, 1]$
- ν πλήθος σημείων (Περισσότερα σημεία,μεγαλύτερη ακρίβεια)

Αντικαθιστώντας την F στην (1.4.2) με συντηρήσεις βάσης B-spline,

$$\tilde{Q}_\nu(B_{j,d}) = \frac{t_{i+1}-t_i}{2} \sum_{a=1}^{\nu} w_a B_{j,d}(\frac{t_{i+1}-t_i}{2} x_a + \frac{t_{i+1}+t_i}{2}) \tag{1.4.3}$$

που είναι το διακριτό ανάλογο με,

$$\tilde{Q}_\nu(B_{j,d}) \approx \int_{t_i}^{t_{i+1}} B_{j,d}(x)dx \tag{1.4.4}$$

καταλήγουμε στην προσέγγιση του ολοκληρώματος της καμπύλης (1.4.1),

$$\int_{t_i}^{t_{i+1}} F(x)dx \approx \sum_{j=0}^{n} c_j \tilde{Q}_\nu(B_{j,d}). \tag{1.4.5}$$

Κάποια από τα βάρη και οι ρίζες δίνονται όπως παρακάτω,

1	0		2	
2	$\pm\frac{1}{\sqrt{3}}$	±0.57735...	1	
3	0		$\frac{8}{9}$	0.888889...
	$\pm\sqrt{\frac{3}{5}}$	±0.774597...	$\frac{5}{9}$	0.555556...
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.339981...	$\frac{18+\sqrt{30}}{36}$	0.652145...
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.861136...	$\frac{18-\sqrt{30}}{36}$	0.347855...

Υλοποίηση του (1.4.2) σε C++ με τις συναρτήσεις βάσης

```
const double x[4] = {-0.8611363115940526, -0.3399810435848563, 0.3399810435848563, 0.8611363115940526};
const double w[4] = {0.34785484513745385, 0.6521451548625461, 0.6521451548625461, 0.34785484513745385};
vector<double> bspline_gauss_legendre(int q,int n,int order,int nk,vector<double> kns ,double (f)(int,int,double,int,vector<
double local_sum = 0; // Sum of a B-spline basis index (ex B1 at [0,1]) in a specific interval
double bspline_sum=0; // Sum of all B-splines basis index(ex B1 at [0,1],[0,2]...) in a all intervals
double total_sum=0; //  Sum of all B-splines basis of every index in all intervals
vector<double> v_bspline_sum; // Vector of sum of all B-splines basis index(ex B1 at [0,1],[0,2]...) in a all intervals
set<pair<double, double>> intervals; // Set of pair of intervals(lower bound,upper bound)
vector<vector<double>> xq;
vector<double> coefficient_intervals;
// Single time claculation for intervals
for (int i = 0; i < kns.size() - 1; i++)
{
  double start = kns[i];
  double end = kns[i+1];
  if (isEqual(start, end))
  {
    continue; // Skip over intervals that start and end with the same value
  }
  intervals.insert(make_pair(start, end));
}
// Single time claculation for quadrature points and interval coefficients and insert into 2d vector(intervals,xi)
for (auto interval : intervals) {
  //cout << interval.first << "-" << interval.second << '\n';
  vector<double> xi;
  for (int i = 0; i < n; i++)
  {
    xi.push_back((interval.second-interval.first)/2 * x[i] + (interval.second+interval.first)/2);
  }
  xq.push_back(xi);
  coefficient_intervals.push_back((interval.second-interval.first)/2);
}
for(int j=0;j<nk-order; j++)
{
  bspline_sum=0; // Reset the bpline sum
  for(int h= 0; h < xq.size(); h++)
  {
    local_sum = 0; //  Reset the local sum
    for (int i = 0; i < xq[h].size() && i < n; ++i)
    {
        local_sum += w[i]*f(j, order-1, xq[h][i], kns.size(), kns);
    }
    local_sum *= coefficient_intervals[h];
    bspline_sum += local_sum;
  }
  v_bspline_sum.push_back(bspline_sum);
}
return v_bspline_sum;
}
```

Κεφάλαιο 2

Προσέγγιση συναρτήσεων

2.1 GNU Scientific Library (GSL)

[1] Είναι μια βιβλιοθήκη προγραμμάτων ανοιχτού κώδικα, υλοποιημένη σε γλώσσα προγραμματισμού C, η οποία προσφέρει τα απαραίτητα εργαλεία για επιστημονικούς υπολογισμούς. Προσφέρει εργαλεία για αριθμητικούς υπολογισμούς στα παρακάτω πεδία:

[4]

Complex Numbers	Roots of Polynomials	Special Functions
Vectors and Matrices	Permutations	Combinations
Sorting	BLAS Support	Linear Algebra
CBLAS Library	Fast Fourier Transforms	Eigensystems
Random Numbers	Quadrature	Random Distributions
Quasi-Random Sequences	Histograms	Statistics
Monte Carlo Integration	N-Tuples	Differential Equations
Simulated Annealing	Numerical Differentiation	Interpolation
Series Acceleration	Chebyshev Approximations	Root-Finding
Discrete Hankel Transforms	Least-Squares Fitting	Minimization
IEEE Floating-Point	Physical Constants	Basis Splines
Wavelets	Sparse BLAS Support	Sparse Linear Algebra

Συγκεκριμένα, θα αξιοποιήσουμε τα πακέτα για Basis splines.

[1] https://www.gnu.org/software/gsl/

Η βιβλιοθήκη προσφέρει και διεπαφές για γλώσσες υψηλού επιπέδου. Για λόγους βελτιστοποίησης, προτιμάται η C.

2.1.1 Χρήση της βιβλιοθήκης

Μεταγλώττιση

Η μεταγλώττιση μπορεί να γίνει με οποιονδήποτε μεταγλωττιστή της C.Παρακάτω χρησιμοποιείται ο μεταγλωττιστής gcc (GNU Compiler).

```
$ gcc -Wall -I/usr/local/include -c example.c
```

- -Wall Ανέφερε όλες τις προειδοποιήσεις και τα λάθη μεταγλώττισης
- -I Τοποθεσία εγκατάστασης της βιβλιοθήκης
- -c Μεταγλώττισε, αλλά μην κάνεις τη σύνδεση

Σύνδεση

Όπως κάθε βιβλιοθήκη στην C, μέτα τη μεταγλώττιση, πρέπει να συνδεθεί με το πρόγραμμα. Η βιβλιοθήκη GSL εμπεριέχεται σε ένα ενιαίο αρχείο στατικής σύνδεσης, libgsl.a[4]. Η βιβλιοθήκη υποστηρίζει και διαμοιραζόμενες βιβλιοθήκες.

```
$ gcc -L/usr/local/lib example.o -lgsl -lgslcblas -lm
```

- -L Τοποθεσία για τις βιβλιοθήκες
- -lgsl Σύνδεση με την βιβλιοθήκη gsl
- -lgslcblas Σύνδεση με τη βιβλιοθήκη CBLAS για υπορουτίνες γραμμικής άλγεβρας.
- -lm Σύνδεση με την υποστηριζόμενη μαθηματική βιβλιοθήκη του συστήματος.

2.2 Διακριτοποίηση

2.2.1 Συναρτήσεις βάσης και διάνυσμα κόμβων

Για να υπολογίσουμε τις συναρτήσεις βάσης, πρέπει πρώτα να δεσμεύσουμε θέσεις στη μνήμη.

```
gsl_bspline_workspace *work = gsl_bspline_alloc(k, nbreak);
```

- $gsl\_bspline\_workspace$ Ειδικά διαμορφωμένη δομή δεδομένων για αποθήκευση B-splines και των ιδιοτήτων τους.
- k τάξη καμπύλης B-spline
- $nbreak$ Το πλήθος των εσωτερικόν κόμβων ,ουσιαστικά εκεί που "σπάει" καμπύλη σε συναρτήσεις βάσης.

Μέσα από τον δείκτη μνήμης $work$ μπορούμε να αντλήσουμε τις παρακάτω πληροφορίες για τις καμπύλες B-spline:

- $work->spline\_order$ τάξη καμπύλης B-spline
- $work->ncontrol$ σημεία ελένχου
- $work->knots$ διάνυσμα κόμβων

Η gsl έχει τη δυνατότητα να αρχικοποιήσει το διάνυσμα κόμβων με διάφορους τρόπους, ώστε να επιτευχθεί η επιθυμητή γεωμετρία μιας καμπύλης B-spline. Η ομοιόμορφη κατανομή έχει τις περισσότερες εφαρμογές, συμπεριλαμβανομένου και του προβλήματος που θα λυθεί παρακάτω. Ένα ομοιόμορφο διάνυσμα κόμβων στο διάστημα $[\alpha, b]$:

$$T = \left\{ \underbrace{a, ..., a}_{d}, t_1, t_2, ..., t_{n-d+1}, \underbrace{b, ..., b}_{d} \right\} \tag{2.2.1}$$

όπου

$$t_i = a + \frac{i-1}{n-d+1}(b-a) \; i = 1, .., n-d+1 \tag{2.2.2}$$

Το μήκος υποδιαστήματος h μεταξύ δύο διαδοχικών διαφορετικών κόμβων του διανύσματος κόμβων T, παράγεται από τη σχέση,

$$h = \frac{(b-a)}{n-d+1} \tag{2.2.3}$$

Παρακάτω στην εργασία για απλότητα θα αναφέρουμε το h ως το μήκος της διαμέρισης του $[a, b]$.
Η εντολή στην GSL είναι:

```
gsl_bspline_init_uniform(a, b, work);
```

Υλοποίηση του (2.2.1) σε C++

```
vector<double> uniform_knot_vector(double n,double d,double a ,double b)
{
    vector<double> knot_vector;
    for (int i=0; i<d; i++)
    {
        knot_vector.push_back(a);
    }
    int i=1;
    for(int g=d; g<n-d+2; g++)
    {
        knot_vector.push_back(a+(double(i-1)/double(n-d))*(b-a));
        i++;
    }
    for(int g=n; g<n+d+1; g++)
    {
        knot_vector.push_back(b);
    }
    return knot_vector;
}
```

2.2.2 Γκραμιανός πίνακας

Έστω $B(x) = (B_{1,d}(x), B_{2,d}(x), ..., B_{n,d})$ είναι ένα διάνυσμα n-διαστάσεων, όπου τα στοιχεία του είναι συναρτήσεις βάσης υπολογισμένες στο σημείο x. Τότε μπορεί να κατασκευαστεί ένας πίνακας A από το τανυστικό γινόμενο του $B(x)$ με τον εαυτό του.

$$A(x) = B(x) \otimes B(x) \tag{2.2.4}$$

Σε μορφή δεικτών:

$$A_{i,j} = B_i(x)B_j(x) \tag{2.2.5}$$

Μπορούμε να γενικεύσουμε την (2.2.4) σε παραγώγους τάξης τ.

$$A^{\tau}_{i,j} = B^{\tau}_i(x)B^{\tau}_j(x) \tag{2.2.6}$$

Έπειτα ολοκληρώνουμε στο διάστημα [a, b] για να αποκτήσουμε έναν n×n Γκραμιανό πίνακα των συναρτήσεων βάσης με παραγώγους τάξης τ.

$$G^{\tau}_{i,j} = \int_a^b A^{\tau}_{i,j}dx = \int_a^b B^{\tau}_i(x)B^{\tau}_j(x)dx \tag{2.2.7}$$

Μπορούμε να δείξουμε τις παρακάτω ιδιότητες,

- Είναι συμμετρικός και θετικά ορισμένος
- Είναι πίνακας ζώνης (banded) με κάτω όριο ζώνης (band) $k-1$,αφού πολλά από τα στοιχεία του είναι μηδενικά, διότι δεν ορίζονται όλες οι συναρτήσεις βάσης στο ίδιο διάστημα.

Για τη δημιουργία του Γκραμιανού πίνακα, έχουμε κατασκευάσει τον κώδικα σε C++,που παρατίθεται παρακάτω.

Υλοποίηση του Γκραμιανού πίνακα σε C++

```
const double x[4] = {-0.8611363115940526, -0.3399810435848563, 0.3399810435848563, 0.8611363115940526};
const double w[4] = {0.34785484513745385, 0.6521451548625461, 0.6521451548625461, 0.34785484513745385};
vector<vector<double>> create_a_deriv_gauss_legendre(int n,int order,int nk,vector<double> kns ,double (f)(int,int,double,in
double local_sum = 0; // Sum of a B-spline basis index (ex B1 at [0,1]) in a specific interval
double bspline_sum=0; // Sum of all B-splines basis index(ex B1 at [0,1],[0,2]...) in a all intervals
double total_sum=0; //  Sum of all B-splines basis of every index in all intervals
vector<double> v_bspline_sum; // Vector of sum of all B-splines basis index(ex B1 at [0,1],[0,2]...) in a all intervals
vector<vector<double>> A;
set<pair<double, double>> intervals; // Set of pair of intervals(lower bound,upper bound)
vector<vector<double>> xq;
vector<double> coefficient_intervals;
// Single time claculation for intervals
for (int i = 0; i < kns.size() - 1; i++)
{
  double start = kns[i];
  double end = kns[i+1];
  if (isEqual(start, end))
  {
    continue; // Skip over intervals that start and end with the same value
  }
  intervals.insert(make_pair(start, end));
}
// Single time claculation for quadrature points and interval coefficients and insert into 2d vector(intervals,xi)
for (auto interval : intervals) {
  //cout << interval.first << "-" << interval.second << '\n';
  vector<double> xi;
  for (int i = 0; i < n; i++)
  {
    xi.push_back((interval.second-interval.first)/2 * x[i] + (interval.second+interval.first)/2);
  }
  xq.push_back(xi);
  coefficient_intervals.push_back((interval.second-interval.first)/2);
}
for(int u=0;u<nk-order; u++)
{
  for(int j=0;j<nk-order; j++)
  {
    bspline_sum=0; // Reset the bpline sum
    for(int h= 0; h < xq.size(); h++)
    {
      local_sum = 0; //  Reset the local sum
      for (int i = 0; i < xq[h].size() && i < n; ++i)
      {
        local_sum += w[i]*(f(u, order, xq[h][i], kns.size(), kns) *
        f(j, order-1, xq[h][i], kns.size(), kns)+f(u, order-1, xq[h][i], kns.size(), kns) *
        f(j, order, xq[h][i], kns.size(), kns));
      }
      local_sum *= coefficient_intervals[h];
      bspline_sum += local_sum;
    }
    v_bspline_sum.push_back(bspline_sum);
  }
  A.push_back(v_bspline_sum); // Some error here
  v_bspline_sum.clear(); // Empty the vector for next iteration
}
return A;
}
```

Υπάρχει έτοιμη συνάρτηση στην GSL όπου κατασκευάζει τον Γκραμιανό πίνακα. Πριν το κάνουμε αυτό, πρέπει να ορίσουμε έναν πίνακα $n \times k$ με τη δομή δεδομένων matrix, η οποία προσφέρεται από τη βιβλιοθήκη.

```
gsl_matrix *G = gsl_matrix_alloc(n, k); /* Gram matrix allocation */
gsl_bspline_gram(1, G, work); /* compute Gram matrix */
```

- 1 τάξη παραγώγου
- G πίνακας αποθήκευσης
- $work$ Δείκτης στον χώρο εργασίας για τις συναρτήσεις βάσης

2.2.3 Προσέγγιση συνάρτησης $f(x)$ με συναρτήσεις βάσης B-splines.

Όπως αναφέρουμε και παρακάτω, θα κατασκευαστεί ένα σύστημα της μορφής $Ax = b$ για να επιτευχθεί η προσέγγιση μιας συνάρτησης.Όπου

- A Γκραμιανός πίνακας
- x Σημεία ελέγχου
- b Δεξιό μέλος

Το δεξιό μέλος b κατασκευάζεται παίρνοντας το ολοκλήρωμα του γινομένου της συνάρτησης (ή της αντίστοιχης μορφής της) που προσπαθούμε να προσεγγίσουμε, με τις συναρτήσεις βάσης,δηλαδή το στοιχείο b_i του διανύσματος b παράγεται από τον υπολογισμό του,

$$b_i = \int f(x) B_{i,d}(x) \tag{2.2.8}$$

```
gsl_bspline_proj_rhs(&F ,y, work);
```

- F Συνάρτηση $f(x)$
- y διάνυσμα αποθήκευσης μεγέθους n

- $work$ Δείκτης στον χώρο εργασίας για τις συναρτήσεις βάσης

Η F και το y έχουν τον δικό τους τύπο δεδομένων που προσφέρεται από την GSL και είναι gsl_function και gsl_vector.

Υλοποίηση του δεξιού μέλους σε C++

```
const double x[4] = {-0.8611363115940526, -0.3399810435848563, 0.3399810435848563, 0.8611363115940526};
const double w[4] = {0.34785484513745385, 0.6521451548625461, 0.6521451548625461, 0.34785484513745385}

vector<double> create_aprox_b_gauss_legendre(int n,int order,int nk,vector<double> kns ,double (f)(int,int,double,int,vector
double local_sum = 0; // Sum of a B-spline basis index (ex B1 at [0,1]) in a specific interval
double bspline_sum=0; // Sum of all B-splines basis index(ex B1 at [0,1],[0,2]...) in a all intervals
double total_sum=0; //  Sum of all B-splines basis of every index in all intervals
vector<double> v_bspline_sum; // Vector of sum of all B-splines basis index(ex B1 at [0,1],[0,2]...) in a all intervals
set<pair<double, double>> intervals; // Set of pair of intervals(lower bound,upper bound)
vector<vector<double>> xq;
vector<double> coefficient_intervals;
// Single time claculation for intervals
for (int i = 0; i < kns.size() - 1; i++)
{
  double start = kns[i];
  double end = kns[i+1];
  if (isEqual(start, end))
  {
    continue; // Skip over intervals that start and end with the same value
  }
  intervals.insert(make_pair(start, end));
}
// Single time claculation for quadrature points and interval coefficients and insert into 2d vector(intervals,xi)
for (auto interval : intervals) {
  //cout << interval.first << "-" << interval.second << '\n';
  vector<double> xi;
  for (int i = 0; i < n; i++)
  {
    xi.push_back((interval.second-interval.first)/2 * x[i] + (interval.second+interval.first)/2);
  }
  xq.push_back(xi);
  coefficient_intervals.push_back((interval.second-interval.first)/2);
}
for(int j=0;j<nk-order; j++) //Do it for n times
{
  bspline_sum=0; // Reset the bpline sum
  for(int h= 0; h < xq.size(); h++)
  {
    local_sum = 0; //  Reset the local sum
    for (int i = 0; i < xq[h].size() && i < n; ++i)
    {
      local_sum += w[i]*(f(j, order, xq[h][i], kns.size(), kns)*g(xq[h][i])+f(j, order, xq[h][i], kns.size(), kns)*g(xq[h][i
      //i++;
    }
    local_sum *= coefficient_intervals[h];
    bspline_sum += local_sum;
  }
  v_bspline_sum.push_back(bspline_sum);

}
return v_bspline_sum;
}
```

Μπορούμε να προσεγγίσουμε μια συνάρτηση f, εφόσον είναι συνεχής στο διάστημα $[a, b]$ με έναν υποχώρο $V = \{B_{1,d}, B_{2,d}, ..., B_{n,d}\}$ των συναρτήσεων βάσης.Θεωρούμε την προβολή της προσέγγισης της $f(x)$ από μια B-spline ,$\Pi f(x)$, η οποία "ζει" στον υποχώρο V και θα έχει την εξής μορφή:

$$\Pi f(x) = c_1 B_{1,d}(x) + c_2 B_{2,d} + ... + c_n B_{n,d} \tag{2.2.9}$$

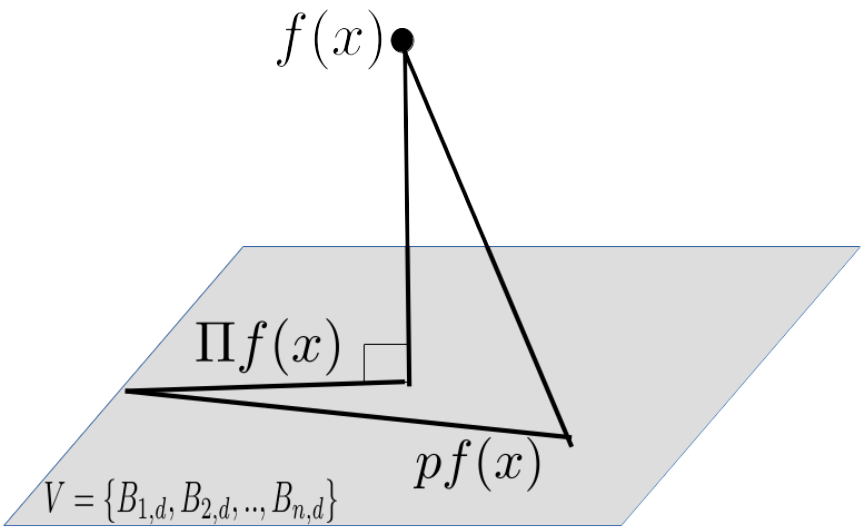

Figure 2.1: Όπου το ορθογώνιο επίπεδο παριστάνει το υποχώρο V

Χρησιμοποιώντας την L_2 προβολή (δηλαδή την βέλτιστη προβολή), θέτουμε το λάθος της προσέγγισης ως κάθετο διάνυσμα στον υποχώρο V, δηλαδή:

$$\int_a^b (\Pi f(x)dx - f(x))B_{i,d}(x) = 0 \qquad \forall\ B_{i,d} \in V \tag{2.2.10}$$

Για να επιτευχθεί η εύρεση των $\mathbf{c} = [c_1, c_2, ..., c_n]$, πρέπει να κατασκευαστεί ένα σύστημα γραμμικών εξισώσεων τύπου $A\mathbf{c} = b$ και να λυθεί.

Το παραπάνω σύστημα, διαλέγοντας διαδοχικά $i = 0, .., n$ και $j = 0, .., n$, μπορεί να προκύψει ένα σύστημα που έχει τη μορφή:

$$\sum_j c_j \underbrace{\int B_{i,d}(x)B_{j,d}(x)dx}_{G_{i,j}} = \underbrace{\int f(x)B_{i,d}(x)dx}_{b_i} \tag{2.2.11}$$

Και σε μορφή πινάκων:

$$A = \begin{bmatrix} \int_a^b B_{0,d}^q(x)B_{0,d}^q(x) & ... & \int_a^b B_{0,d}^q(x)B_{n,d}^q(x) \\ \int_a^b B_{1,d}^q(x)B_{0,d}^q(x) & ... & \int_a^b B_{1,d}^q(x)B_{n,d}^q(x) \\ \vdots & \ddots & \vdots \\ \int_a^b B_{n,d}^q(x)B_{0,d}^q(x) & ... & \int_a^b B_{n,d}^q(x)B_{n,d}^q(x) \end{bmatrix} \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} b = \begin{bmatrix} \int_a^b B_{1,d}f(x) \\ \int_a^b B_{2,d}f(x) \\ \vdots \\ \int_a^b B_{n,d}f(x) \end{bmatrix} \tag{2.2.12}$$

2.2.4 Επίλυση του συστήματος

Η επίλυση του παραπάνω συστήματος αποδίδει τα κατάλληλα σημεία ελέγχου των καμπυλών B-spline για την προσέγγιση της συνάρτησης. Υπάρχουν πολλές μέθοδοι που μπορούν να λύσουν το παραπάνω σύστημα (Gauss, LU, SOR κτλ.).Επειδή ο A είναι θετικά ορισμένος και συμμετρικός και ταινιωτός, επιλέχθηκε η ανάλυση του Cholesky [5], η οποία προσφέρει αριθμητική ευστάθεια και ταχύτητα. Αν και είναι παραλλαγή της ανάλυσης LU, στην προκειμένη περίπτωση είναι περίπου δύο φορές πιο γρήγορη, [6]

Ανάλυση του Cholesky

Αν ο A είναι πραγματικός, θετικά ορισμένος και συμμετρικός, τότε μπορεί να γραφεί ως γινόμενο ενός κάτω τριγωνικού πίνακα με θετικά στοιχεία στη διαγώνιο, με τον ανάστροφό του.

$$A = LL^T \tag{2.2.13}$$

Η επίλυση του συστήματος $A\mathbf{c} = b$ γίνεται αρχικά υπολογίζοντας $Ly = b$ με προς τα μπροστά αντικατάσταση και έπειτα $L^T\mathbf{c} = y$ με προς τα πίσω αντικατάσταση. Η GSL προσφέρει έτοιμες συναρτήσεις για τη μέθοδο Cholesky, ακόμα και στην περίπτωση που ο πίνακας είναι ταινιωτός (Banded).

```
gsl_linalg_cholesky_band_decomp(G);
gsl_linalg_cholesky_band_solve(G, y, c);
```

- G Γκραμιανός πίνακας (A)
- y διάνυσμα αποθήκευσης μεγέθους n (Δεξιό μέλος) (b)
- c διάνυσμα αποθήκευσης μεγέθους n (Σημεία ελέγχου) (c)

Υλοποίηση της Ανάλυση του Cholesky σε C++

```
vector<double> Cholesky_method(const vector<vector<double>> A,const vector<double> b){
    vector<vector<double>> L(A.size(),vector<double> (A.size())),Lt(A.size(),vector<double> (A.size()));
    vector<double> x(L.size(),numeric_limits<double>::quiet_NaN()),y(L.size());
    if(A.size() != A[0].size()){ //Check the dimension of the matrix A and b
        cout << "Not a square matrix";
        return x;}
    vector<vector<double>> At(A.size(),vector<double>(A.size()));
    //Transpose the A into At
    for(int i=0; i<At.size(); i++){
        for(int j=0; j<At[0].size(); j++){
            At[i][j] = A[j][i];}}
    if(A != At) {//Checks for symmetric A ,A=At
        cout << "Not symetric A";
        return x;}
     //Decomposition of A into lower triangular based on the formula (3.24) page 95 Numerical Analysis Dougalis
    double sum=0,sum2=0,sum_x=0,sum_k=0;
    for(int i=0; i<A.size(); i++) {
        for(int j=0; j<i; j++){
            for(int k=0; k<j; k++){
                sum+=L[i][k] * L[j][k];}
            L[i][j]=(A[i][j]-sum)/L[j][j];
            sum=0;}
        for(int g=0; g<i; g++){
            sum2+=pow(L[i][g],2);}
        L[i][i]=sqrt(A[i][i] - sum2);
        if(L[i][i] < 0) {//Check for positive definite of A based on the decomposition of L Lt

            cout << "A is not positive definite" ;
            return x;}
        sum2=0;}
    for(int i=0; i<Lt.size(); i++){ //Build the L transpose Lt
        for(int j=0; j<Lt[0].size(); j++){
            Lt[i][j] = L[j][i]; }}
     for(int i=0; i<L.size(); i++) { //Foward substitution L y= b
        for(int j=0; j<i; j++){
            sum_x+= L[i][j] * y[j];}
        y[i]= (b[i]-sum_x) / L[i][i];
        sum_x=0;}
    x[x.size()- 1] = y[y.size()- 1]/Lt[Lt.size()- 1][Lt.size()- 1];
    for(int k=L.size()-2; k>=0; k--) {//Backwards substitution Lt x = y
        for(int j=k+1; j<=Lt.size()-1; j++){
            sum_k+= Lt[k][j] * x[j];}
        x[k]= (y[k]-sum_k) / Lt[k][k];
        sum_k=0;}
    return x;}
```

2.2.5 Υπολογισμός λύσης

Εφόσον έχουμε βρει τα σημεία ελέγχου c_i , μπορούμε να υπολογίσουμε την B-spline προβολή $\Pi f(x)$.

```
gsl_bspline_calc(x, c, &result, work);
```

- x σημείο υπολογισμού
- c σημεία ελέγχου
- $result$ Μεταβλητή τύπου double για προσωρινή αποθήκευση αποτελέσματος
- $work$ Δείκτης στον χώρο εργασίας για τις συναρτήσεις βάσης

2.2.6 Παραδείγματα

Παρακάτω ορίζουμε B-spline χώρους(δες (1.1.4)) και υπολογίζουμε την βέλτιστη προβολή όπως την περιγράψαμε στο προηγούμενο κεφάλαιο. Ως συνεχή συνάρτηση θεωρούμε την $f(x) = sin(2\pi x)$ και το διάστημα υπολογισμού είναι το $[0, 10]$.

Πρώτο παράδειγμα

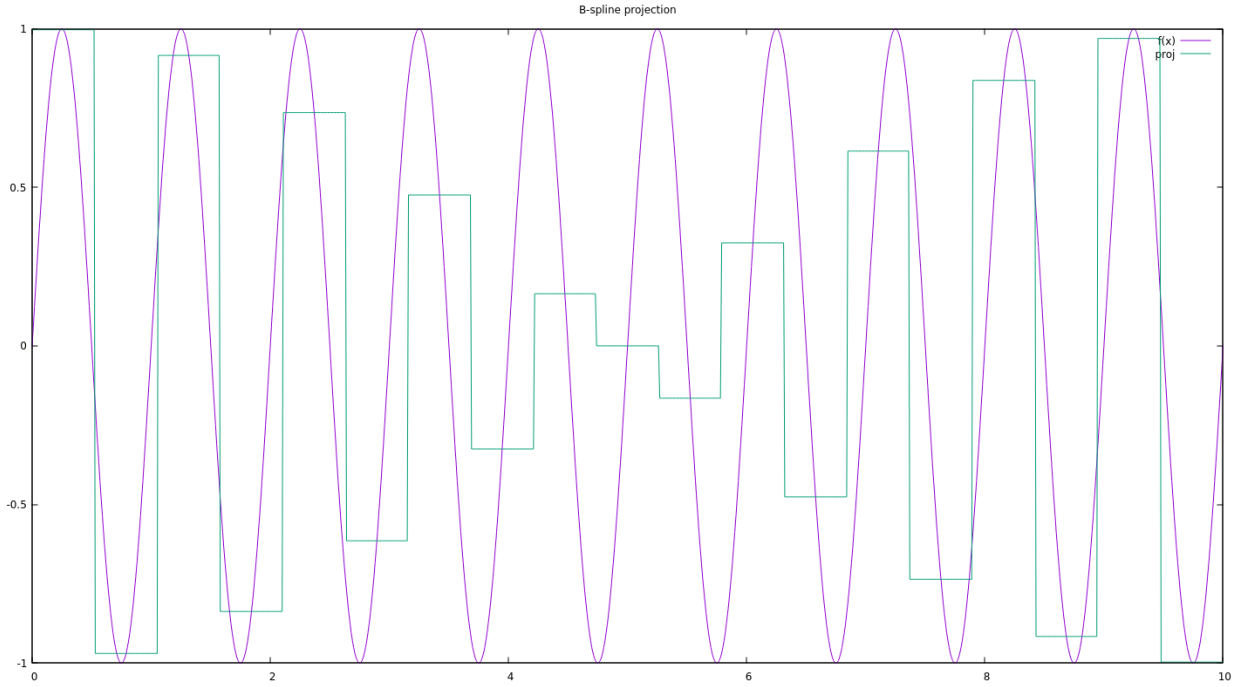

Figure 2.2: Προβολή συνάρτησης $sin(2\pi x)$ με συναρτήσεις βάσης βαθμού 0.

Υπολογίσαμε την βέλτιστη προβολή με συναρτήσεις βάσης μηδενικού βαθμού (όπως φαίνεται από τον 'ψαλιδισμό' της προσέγγισης, τα πολυώνυμα μηδενικού βαθμού αδυνατούν να κατασκευάσουν μια ομαλή καμπύλη), στο διάστημα [0, 10], με ομοιόμορφο διάνυσμα κόμβων από το [0, 10] και 20 εσωτερικούς κόμβους. Στο διάστημα [0,10] και στο γράφημα Figure 2.2 παρουσιάζουμε τις παραστάσεις να είναι αρκετά ανακριβής.

Δέυτερο παράδειγμα

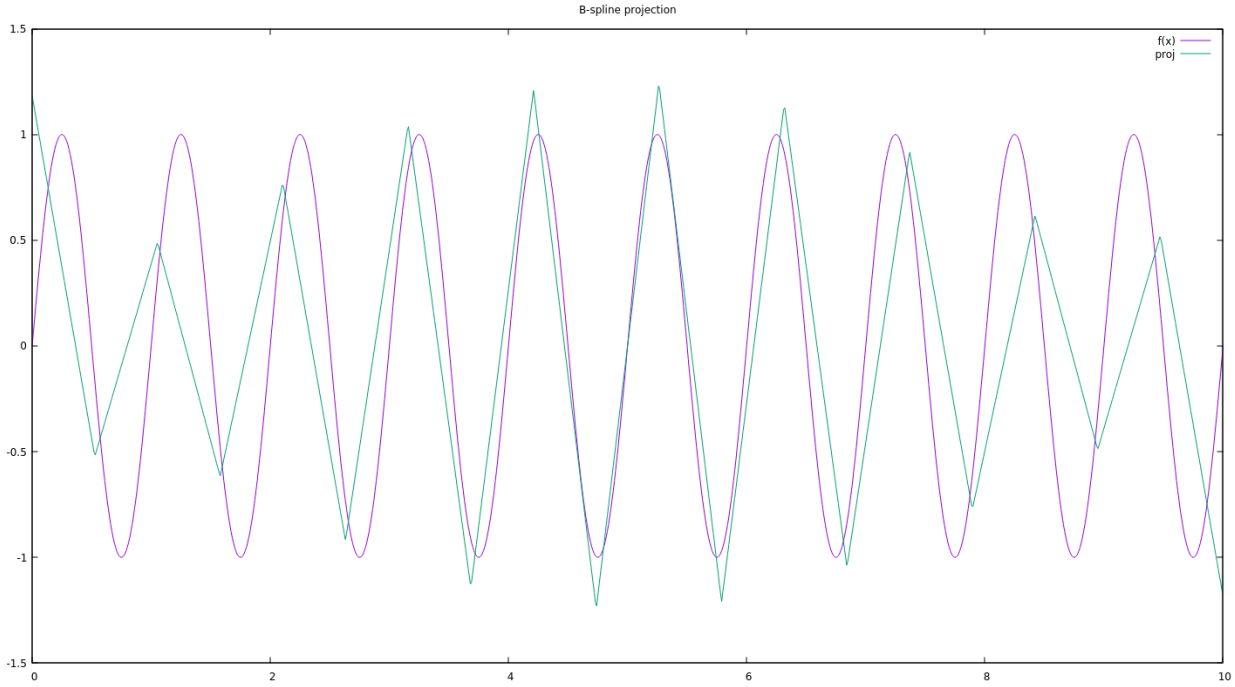

Figure 2.3: Προβολή συνάρτησης $sin(2\pi x)$ με συναρτήσεις βάσης βαθμού 1.

Στο 2.3 διάγραμμα αυξήσαμε τον βαθμό των συναρτήσεων βάσης κατά 1 και κρατήσαμε όλα τα υπόλοιπα ίδια. Εδώ η προσέγγιση είναι πιο ακριβής.

Τρίτο παράδειγμα

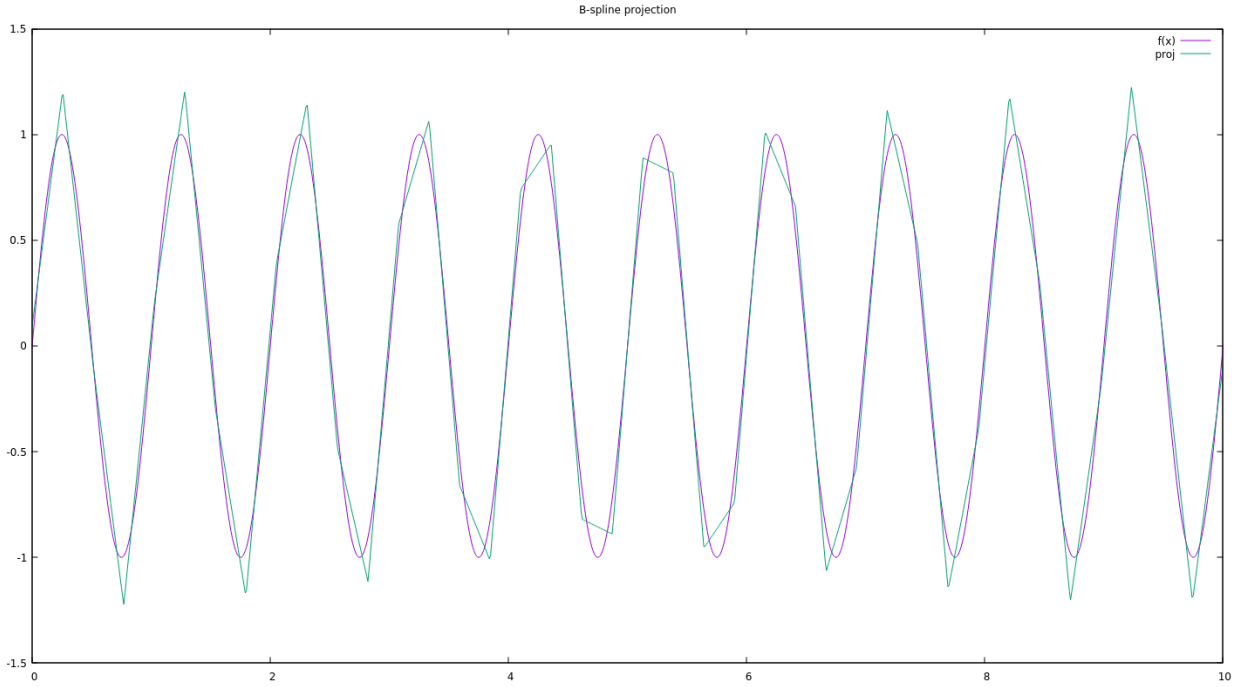

Figure 2.4: Προβολή συνάρτησης $sin(2\pi x)$ με συναρτήσεις βάσης βαθμού 1,με 40 εσωτερικούς κόμβους.

Εδώ, στο 2.4, αυξήσαμε τους εσωτερικούς κόμβους από 20 σε 40. Παρατηρούμε ότι η ακρίβεια αυξήθηκε αρκετά σε σύγκριση με το διάγραμμα 2.3. Όμως, τα πολυώνυμα πρώτου βαθμού εξακολουθούν να μην μπορούν να προσεγγίσουν ομαλά καμπύλες. Το παραπάνω συνεχίζει να ισχύει ακόμη κι αν αυξήσουμε τους εσωτερικούς κόμβους. Δηλαδή, τα πολυώνυμα πρώτου βαθμού είναι ανίκανα να προσεγγίσουν επαρκώς μια καμπύλη.

Τέταρτο παράδειγμα

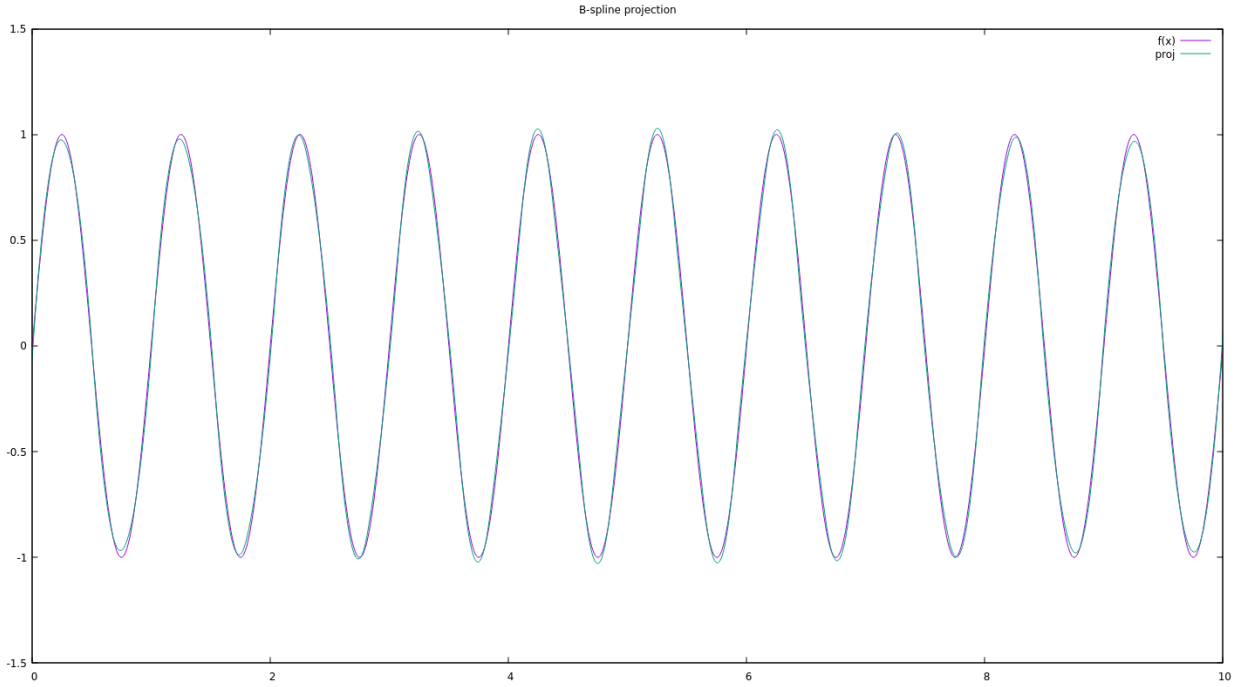

Figure 2.5: Προβολή συνάρτησης $sin(2\pi x)$ με συναρτήσεις βάσης βαθμού 2,με 40 εσωτερικούς κόμβους.

Στο διάγραμμα 2.5 αυξήσαμε τον βαθμό των πολυωνύμων κατά ένα. Η ακρίβεια της προσέγγισης έχει βελτιωθεί σημαντικά, καθώς πλέον τα πολυώνυμα δεύτερου βαθμού έχουν τη δυνατότητα να παράγουν καμπύλες.

2.2.7 Αριθμητικά παραδείγματα

Ένας βέλτιστος πολυωνυμικός χώρος πεπερασμένης διάστασης είναι τα πολυώνυμα Legendre. Παρακάτω παραθέτουμε κάποιους υπολογισμούς για την εύρεση βέλτιστης προσέγγισης με τους δύο αυτούς χώρους. Παίρνοντας την προβολή της συνάρτησης $sin(2\pi x)$ στο διάστημα $[-1, 1]$, υπολογίζουμε τη $L_2 - error$ νόρμα (σφάλμα προσέγγισης $L_2 - error = (\int_a^b (f - \Pi f)^2 dx)^{\frac{1}{2}}$) για διάφορες τιμές του βαθμού των συναρτήσεων βάσης και του διανύσματος κόμβων (ομοιόμορφη κατανομή), σε σχέση με τον χρόνο εκτέλεσης.

Table 2.1: Αποτελέσματα βέλτιστης προσέγγισης με συναρτήσεις βάσης B-spline βαθμού $d = 3$.

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$
0.000072	7.139766e-02	0.500000	3	8	6
0.000012	8.179280e-03	0.250000	3	12	10
0.000021	3.950555e-04	0.125000	3	20	18
0.000024	1.651690e-04	0.100000	3	24	22
0.000042	8.778567e-06	0.050000	3	44	42
0.000075	5.561397e-07	0.025000	3	84	82

Table 2.2: Αποτελέσματα βέλτιστης προσέγγισης με συναρτήσεις βάσης B-spline βαθμού $d = 4$.

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$
0.000125	1.019372e-01	0.500000	4	9	6
0.000075	2.012589e-03	0.250000	4	13	10
0.000057	5.882935e-05	0.125000	4	21	18
0.000063	1.929281e-05	0.100000	4	25	22
0.000091	5.438327e-07	0.050000	4	45	42
0.000149	1.647527e-08	0.025000	4	85	82

- CPU_{time} Η διαφορά του χρόνου από την αρχή της εκτέλεσης του προγράμματος μέχρι τη λήξη του. Το πρόγραμμα χρονομετρήθηκε με τη συνάρτηση *clock()* από τη βιβλιοθήκη *time.h* και εκφράζεται σε δευτερόλεπτα. Η χρονομέτρηση έγινε σε έναν προσωπικό υπολογιστή με επεξεργαστή Intel® Core™ i7-2640M $\times$ 2 και μνήμη 8.0 DDR3.
- $nbreak$ Το πλήθος των εσωτερικόν κόμβων όπου η σχέση με το μήκος διαμέρισης, h είναι $h = \frac{(b-a)}{nbreak}$.

Μεγάλου βαθμού πολυώνυμα, υπολογισμένα σε λίγα υποδιαστήματα, έχουν μεγαλύτερο υπολογιστικό κόστος από ό,τι χαμηλού βαθμού πολυώνυμα σε πολλά υποδιαστήματα, με αντάλλαγμα χαμηλότερο σφάλμα.

Σύγκριση με πολυώνυμα Legendre

Κατασκευάζοντας το ίδιο σύστημα εξισώσεων ($Ax = b$) με τη διαφορά ότι, αντί για συναρτήσεις βάσης, χρησιμοποιούμε τα πολυώνυμα Legendre και λύνοντάς το με τον ίδιο τρόπο και για τα ίδια δεδομένα όπως παραπάνω, προχωράμε στη σύγκρισή τους ως προς την ακρίβεια της προσέγγισης και τον χρόνο εκτέλεσής τους.

Table 2.3: Αποτελέσματα βέλτιστης προσέγγισης με πολυώνυμα Legendre [2]

P_d	$L_2 - error$	CPU_{time}
3	1.0180875411	0.0000600000
4	1.0180875411	0.0000730000
5	0.5040636619	0.0001080000
6	0.5040636619	0.0001140000
7	0.0805605009	0.0001350000
8	0.0805605009	0.0001530000

Table 2.4: Αποτελέσματα βέλτιστης προσέγγισης με συναρτήσεις βάσης B-spline

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$
0.000069	1.075045e+00	2.000000	3	5	3
0.000006	2.271086e-01	1.000000	3	6	4
0.000006	1.030563e-01	0.500000	3	8	6
0.000013	1.419341e-01	0.500000	4	9	6
0.000011	1.607406e-02	0.400000	4	10	7

[2] Η επαναλαμβανόμενη φύση των αποτελεσμάτων οφείλεται στην ιδιότητα της $sin(2\pi x)$ ότι είναι περιοδική.

Κεφάλαιο 3

Πρόβλημα διάχυσης

Όπως αναφέραμε και στην αρχή της εργασίας, η μαθηματική μοντελοποίηση διαφόρων φυσικών προβλημάτων οδηγεί σε μερικές διαφορικές εξισώσεις, όπου σπάνια είναι εύκολο να βρεθεί η αναλυτική τους λύση. Έτσι, καταφεύγουμε σε αριθμητικές μεθόδους για να προσεγγίσουμε τις λύσεις. Λύνοντας το πρόβλημα αριθμητικά, υπάρχει πάντα ένα σφάλμα προσέγγισης, το οποίο προσπαθούμε να περιορίσουμε ανάλογα με την ακρίβεια που θέλουμε να επιτύχουμε. Το πρόβλημα που θα επιλύσουμε είναι το γνωστό πρόβλημα διάχυσης-αντίδρασης σε ένα φραγμένο διάστημα $[a, b]$. Το πρόβλημα γράφεται ως:

$$\begin{cases} -u''(x) + qu(x) = f(x) \quad \text{στο} [a, b], \\ u(a) = u_a \quad u(b) = u_b \end{cases} \tag{3.0.1}$$

Το (3.0.1) είναι ένα πρόβλημα δύο σημείων με συνοριακές συνθήκες Dirichlet, και με $q > 0, f \in C[a, b], a < b \quad u_a, u_b \in \mathbb{R}$. Για παράδειγμα αν το (3.0.1) έχει πραγματική λύση τέτοια ώστε $u(a) = u(b) = 0$, το πρόβλημα έχει ομογενείς συνθήκες Dirichlet ,δηλαδή $u(a = 0) = u_a = u(b = 1) = u_b = 0$.

3.0.1 Μέθοδος Galerkin με γενικές B-splines

Η κύρια ιδέα αυτής της μεθόδου είναι να διακριτοποιήσουμε ένα συνεχές πρόβλημα με μεθόδους πεπερασμένων στοιχείων [7], τα οποία οδηγούν σε έναν απλό αλγόριθμο δηλαδή έυκολα να υλοποιηθεί στον υπολογιστή και έτσι είναι εύκολο να υπολογιστεί η αριθμητική-προσεγγιστική λύση.
Στο συγκεκριμένο κεφάλαιο θα αναπτυχθεί ενδελεχώς η μεθοδολογία και οι τεχνικές επίλυσης προβλημάτων αριθμητικής ανάλυσης με τη χρήση της μεθόδου των πεπερασμένων στοιχείων για την επίλυση διαφορικών εξισώσεων. Η βασική διαδικασία είναι σχετικά παρόμοια για προβλήματα μίας, δύο και τριών διαστάσεων, τα οποία θα παρουσιαστούν στις επόμενες παραγράφους του κεφαλαίου. Τα κυριότερα βήματα για την επίλυση προβλημάτων με τη βοήθεια της μεθόδου των πεπερασμένων στοιχείων είναι τα παρακάτω [8] :

- Ο προσδιορισμός της ισχυρής μορφής της διαφορικής εξίσωσης.
- Ο προσδιορισμός της ασθενούς μορφής της διαφορική εξίσωσης.
- Η επιλογή των κατάλληλων συναρτήσεων βάσης και προσεγγιστικών συναρτήσεων για το άγνωστο πεδίο.
- Η επίλυση του συστήματος εξισώσεων πεπερασμένων στοιχείων.

Στη παρούσα εργασία παρουσιάζεται η διαδικασία επίλυσης με τέτοιο τρόπο έτσι ώστε να αντιστοιχεί στα προαναφερθέντα βήματα. Στο πρώτο μέρος της διαδικασίας ορίζεται το πρόβλημα με τη βοήθεια της ισχυρής μορφής και προσδιορίζονται οι συνοριακές συνθήκες. Έπειτα μελετάται η ασθενής μορφή και αποδεικνύεται η ισοδυναμία με την αντίστοιχη ισχυρή μορφή. Τέλος, ακολουθεί η προσέγγιση του άγνωστου πεδίου, όπου είναι και η λύση των εξισώσεων πεπερασμένων στοιχείων. Επιπλέον, εξηγείται η έννοια της προσεγγιστικής συνάρτησης στα πλαίσια της μεθόδου των πεπερασμένων στοιχείων, οι συναρτήσεις βάσης των B-spline (δες (1.1.3)), και πώς χρησιμοποιούνται στον ορισμό της ασθενούς μορφής της διαφορικής εξίσωσης του προβλήματος.

Ασθενής μορφή

Η ασθενής διατύπωση μιας διαφορικής εξίσωσης μας επιτρέπει τη εκφρασή της σαν μια σχέση μεταβολών, δηλαδή μια σχέση ολοκληρωμάτων.

Η έκφραση της (3.0.1) στην ασθενή μορφή της με συναρτήσεις βάσης B-spline είναι,

$$\int_a^b (u'(x)B'_{i,d}(x)+q(x)u(x)B_{i,d}(x)\,dx = \int_a^b fB_{i,d}\,dx-(u'(a)B_{i,d}(a)+u'(b)B_{i,d}(b)) \tag{3.0.2}$$

Θέλουμε να εκφράσουμε το διακριτό ανάλογο της (3.0.2) μέσα στους χώρους των B-splines που μελετήσαμε προηγουμένως (δίνεται στην (1.1.4)) που περιέχει τις συναρτήσεις βάσης B-spline και την u_h, οι ιδιότητές του αναφέρονται στο Κεφάλαιο 1.3. Ο διαμερισμός που επιλέξαμε για τον $I = [a, b]$ είναι ομοιόμορφος και ορίζεται ως,

$$\begin{aligned} D = \{a = x_0 < x_1 < ... < x_{n+1} = b\} \text{ με } n := (b-a)/h + 1 \\ \text{και } h := x_{i+1} - x_i \quad i = 0, ..., n \end{aligned} \tag{3.0.3}$$

Απο τον διαμερισμό (3.0.3) προκύπτει το διάνυσμα κόμβων T (δες (1.1.1)). Κάνοντας χρήση του διανύσματος T ορίζουμε τον χωρο V_h των B-splines, μεσα στο χωρο V_h η προσεγγιστική λύση έχει μορφή,

$$u_h(x) = \sum_{i=1}^{n} c_i B_{i,d}(x), \tag{3.0.4}$$

$$u(x) \approx u_h(x) \tag{3.0.5}$$

και για να την βρούμε πρέπει να υπολογίσουμε τους συντελεστές $c_1, c_2, .., c_n$. Ουσιαστικά αυτό θα το κάνει ο υπολογιστής με το πρόγραμμα που κατασκευάσαμε.

Σύστημα εξισώσεων

Αντικαθιστώντας στην (3.0.2) όπου u την προσεγγιστική λύση u_h προκύπτει η διακριτή μορφή της (3.0.2) στο χώρο V_h,

$$\int_a^b -u_h' B_{i,d}'\, dx + q\int_a^b u_h B_{i,d}\, dx = \int_a^b f B_{i,d}\, dx + [u_h'(b)B_{i,d}(b) + u_h'(a)B_{i,d}(a)] \tag{3.0.6}$$

όπου η (3.0.6) προέκυψε απο εφαρμογή του κανόνα ολοκλήρωσης κατά μερή. Απο την (3.0.4) και την (3.0.6) μπορούμε να οδηγηθούμε σε ενα σύστημα γραμμικών εξισώσεων με αγνωστους τα $C = [c_1, , c_2.., c_n]$. Το σύστημα θα έχει τη μορφή $AC = F$, οπού ο $A = S + M$ αποτελείται από τον πίνακα ακαμψίας S και τον πίνακα μάζας M.

Ο πίνακας ακαμψίας είναι, ουσιαστικά, ο Γκραμιανός πίνακας που ορίσαμε στα προηγούμενα κεφάλαια, (2.2.7), για $\tau = 1$, με διαστάσεις $n \times n$ και έχει στοιχεία,

$$S_{i,j} = \int_a^b B_{i,d}'(x)B_{i,d}'(x)\, dx = G_{i,j}'. \tag{3.0.7}$$

Ο πίνακας μάζας είναι, ο Γκραμιανός πίνακας με διαστάσεις $n \times n$, για $\tau = 0$, και κάθε στοιχείο του πολλαπλασιάζεται με τη σταθερά q,

$$M_{i,j} = \int_a^b B_{i,d}(x)B_{i,d}(x)\, dx = G_{i,j} \quad q > 0 \text{ σταθερά.} \tag{3.0.8}$$

Το πρόβλημα (3.0.1) περιέχει και περιορισμούς στα άκρα του διαστήματος, δηλαδή συνοριακές συνθήκες. Έτσι, πρέπει να κατασκευάσουμε δύο πίνακες BCa, BCb με διαστάσεις $n \times n$, υπολογισμένους στα άκρα του διαστήματος όπου τα στοιχεία τους έχουν μορφή,

$$\begin{aligned} BCa_{ij} =& B_{i,d}(a)B_{j,d}(a), \\ BCb_{ij} =& B_{i,d}(b)B_{j,d}(b), \quad i,j = 1,...,n. \end{aligned} \tag{3.0.9}$$

Έχοντας τις ιδιότητες των συναρτήσεων βάσης (δες ενότητα 1.1.3), συγκεκριμένα την ιδιότητα του τοπικής υποστηρίξεις (Local Support), οι πίνακες

BCa και BCb θα είναι αραιοί. Πολλαπλασιάζοντας τον καθένα από τους παραπάνω πίνακες με έναν μεγάλο αριθμό (π.χ. $\tilde{k} = 10^8$), καταλήγουμε να εφαρμόζουμε ένα είδος 'ποινής' στα στοιχεία των πινάκων και στην επιβολή των συνοριακών συνθηκών. Έχοντας υπολογίσει τις συνοριακές συνθήκες στο αριστερό μέλος, πρέπει να κάνουμε το αντίστοιχο και για το δεξί μέλος, κατασκευάζοντας δύο νέα διανύσματα στήλης $n \times 1$,

$$Ba_i = u_a(a)B_{i,d}(a) \quad i = 1, ..., n \tag{3.0.10}$$

$$Bb_i = u_b(b)B_{i,d}(b) \quad i = 1, ..., n. \tag{3.0.11}$$

Οπότε, ο A και, κατά επέκταση, το αριστερό μέλος των εξισώσεων γίνεται,

$$A = S + M + \tilde{k}(BCa + BCb) \tag{3.0.12}$$

και αντίστοιχα στην κατασκευή του δεξιού μέλους των εξισώσεων, παίρνουμε το ολοκλήρωμα της f πολλαπλασιάζοντάς το με τις συναρτήσεις βάσης, καταλήγοντας σε ένα διάνυσμα στήλη $n \times 1$,

$$F_i = \int_a^b f(x)B_{i,d}(x)\ dx \tag{3.0.13}$$

Προσθέτοντας και τις συνοριακές συνθήκες με την ποινή, το F και, κατ' επέκταση, το δεξί μέλος των εξισώσεων είναι:

$$F + \tilde{k}(Ba + Bb) \tag{3.0.14}$$

Άρα τελικά το σύστημα που λύνουμε για την εύρεση του $C = [c_1, c_2, ..., c_n]$ έχει τη μορφή

$$AC = (S + M\tilde{k}(BCa + BCb))C = F + \tilde{k}(Ba + Bb) \tag{3.0.15}$$

Επίλυση του συστήματος

Η επίλυση του παραπάνω συστήματος (3.0.15) μας δίνει κατάλληλα σημεία ελέγχου των καμπυλών B-spline για την εύρεση της αριθμητικής λύσης u_h. Το σύστημα επιλύθηκε με τη μέθοδο Cholesky, αφού ο πίνακας A έχει τις ίδιες ιδιότητες όπως στο προηγούμενο κεφάλαιο.

3.1 Αριθμητικά αποτελέσματα

3.1.1 Πρόβλημα 1

Θεωρούμε την εξίσωση (3.0.1) στο διάστημα $[a = 0, b = 10]$ με ομογενείς συνοριακές συνθήκες $u_a = u_b = 0$, $q = 1$ και με πραγματική λύση $u(x) = sin(2\pi x)$ (οπότε η f στο δεξιό ορίζεται κατάλληλα). Για την επίλυση του προβλήματος (3.0.15) και (3.0.6) επιλέξαμε διαδοχικά το μήκος h του υποδιαστήματος μεταξύ των κόμβων όπως φαίνεται στην πρώτη στήλη του πίνακα Table 3.1 και ο βαθμός των B-spline δίνεται στην δεύτερη στήλη του πινάκα Table 3.1. Στην τρίτη στήλη φαίνεται το πλήθος των εσωτερικών κόμβων. Το συνολικό πλήθος των υποδιαστημάτων είναι $nbreak+2$ γιατί είναι ομοιόμορφη κατανομή κόμβων στο διάνυσμα T (δες (2.2.1)). Στην τέταρτη στήλη φαίνεται η L_2 νόρμα σφάλματος (δες σελίδα 39) . Στην τελευταία στήλη φαίνεται ο χρόνος εκτέλεσης του προγράμματος (δες σελίδα 39).

Table 3.1: Αποτελέσματα: Μέθοδος Galerkin με γενικές B-splines διαφόρου βαθμού

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$
0.001248	5.650578e+00	1.000000	7	18	12
0.001240	2.524146e+00	0.500000	6	27	22
0.001186	2.255231e-02	0.200000	3	54	52
0.005489	1.766971e-10	0.100000	10	111	102
0.003412	2.769526e-06	0.050000	4	205	202
0.004919	6.553341e-05	0.020000	2	503	502

Αξιολογώντας την ελάττωση του L_2 σφάλματος και την διακύμανση του CPU_{time} συμπεράνουμε ότι πιο αποτελεσματικός φαίνεται ο υπολογισμός οπού χρησιμοποιήθηκαν υψηλού βαθμού συναρτήσεις βάσεις B-spline με λιγότερα υποδιαστήματα διαμέρισης του $[a, b]$. Οι παρακάτω πινάκες παραθέτουν παρόμοια αποτελέσματα με τον προηγούμενο πινάκα με την διαφορά ότι ο βαθμός B-spline παραμένει σταθερός. Υπολογίζουμε ακόμα τον ρυθμό σύγκλισης r_i του σφάλματος μεταξύ των διαδοχικών διαμερίσεων του διαστήματος $[a, b]$ με μικρότερο μήκος h_i. Ο ρυθμός r_i υπολογίζεται από τον τύπο,

$$r_i = \frac{log(\frac{e_i}{e_{i+1}})}{log(\frac{h_i}{h_{i+1}})} \tag{3.1.1}$$

- e_i η L_2 νόρμα σφάλματος, $e_i = (\int_a^b (u(x) - u_h(x))^2 \, dx)^{\frac{1}{2}}$
- h το μήκος διαμέρισης του $[a, b]$ (μεγίστη απόσταση μεταξύ δυο διαδοχικών διαφορετικών κόμβων του T, δες (2.2.3))

Table 3.2: Αποτελέσματα: Μέθοδος Galerkin με γενικές B-splines βαθμού $d = 2$

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$	r_i
0.000974	7.106567e+00	1.000000	2	13	12	
						1.034838
0.000971	3.468507e+00	0.500000	2	23	22	
						3.905104
0.001290	9.686020e-02	0.200000	2	53	52	
						3.412332
0.001743	9.097691e-03	0.100000	2	103	102	
						3.089147
0.002493	1.069068e-03	0.050000	2	203	202	
						3.047049
0.005242	6.553341e-05	0.020000	2	503	502	
						3.047742
0.009817	7.925031e-06	0.010000	2	1003	1002	
						3.003425
0.091971	7.862783e-09	0.001000	2	10003	10002	

Table 3.3: Αποτελέσματα: Μέθοδος Galerkin με γενικές B-splines βαθμού $d = 3$

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$	r_i
0.000449	6.270819e+00	1.000000	3	14	12	
						1.228692
0.000432	2.675784e+00	0.500000	3	24	22	
						5.213404
0.000594	2.253352e-02	0.200000	3	54	52	
						4.537427
0.000843	9.703479e-04	0.100000	3	104	102	
						4.116085
0.001447	5.595804e-05	0.050000	3	204	202	
						4.189441
0.002961	1.204249e-06	0.020000	3	504	502	
						3.879893
0.005957	8.179979e-08	0.010000	3	1004	1002	
						4.003891
0.053234	8.107011e-12	0.001000	3	10004	10002	

3.1.2 Πρόβλημα 2

Παρακάτω δίνουμε ένα ακόμα αριθμητικό παράδειγμα της εξίσωσης (3.0.1) στο διάστημα $[a = 0, b = 10]$ με μη ομογενείς συνοριακές συνθήκες $u_a = 4$ $u_b = 16$, $q = 1$ και με πραγματική λύση $u(x) = (x - 2)^2$ (οπότε η f στο δεξιό ορίζεται κατάλληλα). Ο τρόπος παρουσίασης των αποτελεσμάτων είναι όμοιος με το προηγούμενο παράδειγμα.

Table 3.4: Αποτελέσματα: Μέθοδος Galerkin με γενικές B-splines βαθμού $d = 1$

CPU_{time}	$L_2 - error$	h	d	n	$nbreak$	r_i
0.000848	2.372607e-01	1.000000	1	12	12	
						1.794018
0.000069	6.841835e-02	0.500000	1	22	22	
						2.002140
0.000131	1.092549e-02	0.200000	1	52	52	
						1.965313
0.000246	2.797838e-03	0.100000	1	102	102	
						1.959205
0.000484	7.195203e-04	0.050000	1	202	202	
						2.051479
0.001004	1.735747e-04	0.025000	1	402	402	
						1.910716
0.002345	3.013952e-05	0.010000	1	1002	1002	
						2.057755
0.025830	2.638647e-07	0.001000	1	10002	10002	

3.1.3 Υπολογισμός σφάλματος

Αρχικά, παρατηρούμε ότι όσο το h τείνει προς το 0,το L^2 σφάλμα τείνει στο μηδέν και έτσι η προσεγγιστική λύση τείνει προς την πραγματική. Φυσικά, σε ένα πεπερασμένο σύστημα, όπως ένας υπολογιστής, δεν είναι δυνατόν να ορίσουμε έναν αριθμό να τείνει στο άπειρο. Για αυτόν τον λόγο υπάρχει ένα σφάλμα. Το ίδιο ισχύει και για την αριθμητική υπολογισμό ολοκληρωμάτων. Τα δύο αυτά σφάλματα μαζί διαμορφώνουν το συνολικό σφάλμα διακριτοποίησης.

3.2 Παράρτημα

Στην (3.0.1), που ονομάζεται και 'δυνατή' μορφή του προβλήματος, συμβαίνει συχνά να μην υπάρχουν λύσεις ή να μην είναι καλά ορισμένες ,[9]. Για αυτό καταλήγουμε σε κάτι που ονομάζουμε 'ασθενή' μορφή του προβλήματος. Η ονομασία αυτή προέρχεται από την ιδέα ότι αναζητούμε 'ασθενέστερες' μορφές της λύσης, δηλαδή λύσεις με λιγότερο αυστηρές προϋποθέσεις σε σχέση με τη δυνατή μορφή, όπου οι προϋποθέσεις είναι αυστηρές.

Βιβλιογραφία

[1] Pawan Gami. Nurbs calculator. https://nurbscalculator.in/, 2016.

[2] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, 2001.

[3] Γεώργιος Ακρίβης Βασίλειος Δουγαλής. *Εισαγωγή στην Αριθμητική Ανάλυση*. Πανεπιστημιακές Εκδόσεις Κρήτης., 2005.

[4] Mark Galassi. *GNU Scientific Library: Reference Manual*. gnu, 2021.

[5] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

[6] William Press Saul Teukolsky William Vetterling Brian Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University England EPress, 1992.

[7] Ionut Danaila Pascal Joly Sidi Mahmoud Kaber Marie Postel. *An Introduction to Scientific Computing*. Springer, 2007.

[8] Thomas Joseph Robert Hughes. *The Finite Element Method – Linear Static Dynamic Finite Element Analysis*. Prentic-Hall, 1987.

[9] Yair Shapira. *Solving PDEs in C++*. SIAM, 2006.