ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τμηματικά Νευρωνικά Δίκτυα για Προσαρμογή Δεδομένων

Ιωάννης Τσούλος
26 Σεπτεμβρίου, 2001

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Ευχαριστίες

Πριν ξεκινήσει η παρουσίαση της εργασίας αυτής θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κύριο Ισαάκ Λαγαρή, τον επίκουρο καθηγητή κύριο Αριστείδη Λύκα, τον διδάκτορα κύριο Δημήτρη Παπαγεωργίου και τον συνάδερφο κύριο Φώτη Θέο, χωρίς την συνεισφορά των οποίων δεν θα μπορούσε να είχε πραγματοποιηθεί η εργασία που παρουσιάζεται σε αυτό το κείμενο.

1.2 Εισαγωγή

Στο κείμενο αυτό θα συζητηθούν και θα αναλυθούν ζητήματα, όπως η **προσαρμογή δεδομένων**, τα **τεχνητά νευρωνικά δίκτυα**, ο **παράλληλος προγραμματισμός**. Με τον όρο **προσαρμογή δεδομένων** εννοούμε την εύρεση αναλυτικής μορφής μίας συναρτήσεως που να ικανοποιεί μία σειρά από δεδομένα που διαθέτουμε από παρατηρήσεις. Αν βρούμε μία τέτοια μορφή θα μπορέσουμε να προβλέψουμε την τιμή που θα επέστρεφαν οι παρατηρήσεις μας αν γίνοταν σε διαφορετικά σημεία από αυτά των προηγουμένων μετρήσεων, δηλαδή θα περάσουμε από την παρατήρηση στην πρόβλεψη. Η σειρά των μεθόδων που αναπτύσσονται στον τομέα της **προσαρμογής δεδομένων** μπορούν να μας βοηθήσουν σε πολλές επιστημονικές περιοχές, όπως Ιατρική, Χημεία κτλ. Τα **τεχνητά νευρωνικά δίκτυα** μπορούν να θεωρηθούν σαν μία απεικόνιση των διασυνδέσεων των νευρώνων του ανθρωπίνου εγκεφάλου και θεωρούνται σαν μια ικανοποιητική μεθοδολογία στον χώρο της τεχνητής νοημοσύνης, καθώς έχουν την τάση να μαθαίνουν μέσα από παραδείγματα με τον ίδιο περίπου τρόπο που πιστεύεται πως μαθαίνει ο ανθρώπινος εγκέφαλος. Τέλος ο **παράλληλος προγραμματισμός** μπορεί να θεωρηθεί σαν την διαδικασία χρήσεως πολλών επεξεργαστών ταυτόχρονα για την επίλυση δύσκολων και εξαιρετικά χρονοβόρων προγραμμάτων. Με την πτώση στις τιμές των επεξεργαστών έχει ανοίξει ο δρόμος για την ανάπτυξη προγραμματιστικών βιβλιοθηκών που θα βοηθούν πολλές υπολογιστικές μονάδες στην από κοινού επίλυση δύσκολων προβλημάτων.

1.3 Στόχος της εργασίας

Στην εργασία που παρουσιάζεται στην συνέχεια γίνεται μία προσπάθεια να χρησιμοποιηθούν τεχνικές από τους χώρους των **τεχνητών νευρωνικών δικτύων** και του **παράλληλου προγραμματισμού** για την επίλυση προβλημάτων **προσαρμογής δεδομένων**. Τα **τεχνητά νευρωνικά δίκτυα** σήμερα χρησιμοποιούνται με επιτυχία σε δύο μεγάλες κατηγορίες: την διάκριση κατηγοριών και το regression, δηλαδή της προσεγγίσεως αγνώστων συναρτήσεων. Η επιτυχία των **τεχνητών νευρωνικών δικτύων** στην δεύτερη κατηγορία προβλημάτων καθιστά τα **τεχνητά νευρωνικά δίκτυα** κατάλληλα για την αντιμετώπιση των προβλημάτων που πραγματεύεται η **προσαρμογή δεδομένων**. Για να μπορέσουν τα **τεχνητά νευρωνικά δίκτυα** να επιτύχουν καλά αποτελέσματα στην **προσαρμογή δεδομένων**, δηλαδή για να μάθουν μία συνάρτηση, απαιτούν μία σειρά από δεδομένα εκπαιδεύσεως και προσπαθούν με διάφορες τεχνικές να μάθουν την συμπεριφορά της άγνωστης συναρτήσεως ακόμα και σε σημεία που δεν υπάρχουν στα δεδομένα εκπαιδεύσεως. Αυτές οι τεχνικές ξεκινούν από ένα σχετικά μεγάλο σφάλμα προσεγγίσεως της άγνωστης συναρτήσεως και σταδιακά προσπαθούν να το ελαττώσουν. Ωστόσο αυτές οι τεχνικές συνήθως είναι αρκετά χρονοβόρες χωρίς να εγγυώνται πάντοτε ένα καλό αποτέλεσμα. Έτσι πρέπει να αναζητηθούν τρόποι για την μείωση του απαιτούμενου χρόνου. Αυτό που μπορεί να γίνει είναι να επαναδιατυπωθεί το πρόβλημα της **προσαρμογής δεδομένων** με τέτοιο τρόπο ώστε να μπορεί να επιλυθεί με την χρήση μεθόδων **παράλληλου προγραμματισμού**. Στην εργασία αυτή διαμερίσαμε προβλήματα **προσαρμογής δεδομένων** σε πολλά μικρότερα και χρησιμοποιήσαμε μοντέλα με **νευρωνικά δίκτυα** σε κάθε επιμέρους σύνολο δεδομένων τα οποία και εκπαιδεύτηκαν παράλληλα. Επειδή κάτι τέτοιο από μόνο του δεν είναι αρκετό χρησιμοποιήσαμε κατάλληλες συνθήκες στα σημεία γειτνιάσεως των επιμέρους συνόλου δεδομένων ώστε ξεχωριστά μοντέλα να δίνουν την ίδια τιμή σε αυτά τα σημεία, δηλαδή επιβάλλαμε κάποιες συνθήκες συνέχειας.

1.4 Σχετική Εργασία

Για το θέμα της προσεγγίσεως συναρτήσεων με τμηματικά νευρωνικά δίκτυα δεν υπάρχει κάποια σχετική (δημοσιευμένη) εργασία που να είναι υπόψιν μας. Ωστόσο στο θέμα της προσεγγίσεως συναρτήσεων υπάρχουν αρκετές εργασίες. Μία εξ αυτών είναι η μέθοδος της παρεμβολής, με την οποία προσεγγίζουμε τα δεδομένα με την χρήση πολυωνύμων. Περισσότερα για την μέθοδο της παρεμβολής θα δούμε στο δεύτερο κεφάλαιο. Μία ακόμα τεχνική είναι η χρήση ελαχίστων τετραγώνων σαν εκτιμητής της μέγιστης πιθανοφάνειας. Περισσότερα για αυτό μπορεί κανείς να δει στο [2]. Τέλος μία ακόμα κατηγορία τεχνικών που υπάρχει στην βιβλιογραφία είναι η χρήση τεχνικών ακτίνων εμπιστοσύνης για τις παραμέτρους του μοντέλου που προσπαθεί να εκτιμήσει την αντικειμενική συνάρτηση. Κυριότερος εκπρόσωπος αυτής της κατηγορίας τεχνικών είναι η μέθοδος Levenberg για την οποία θα μιλήσουμε στο κεφάλαιο 3 που ασχολείται με την βελτιστοποίηση.

1.5 Συνοπτική περιγραφή των περιεχομένων

Το κείμενο της εργασίας έχει την ακόλουθη δομή:

- **Κεφάλαιο 2 - Παρεμβολή**. Είναι μία από τις βασικότερες τεχνικές **προσαρμογής δεδομένων**. Σε αυτό το κεφάλαιο μελετώνται οι κυριότερες μέθοδοι παρεμβολής και παρουσιάζονται κάποια πειραματικά αποτελέσματα από την εφαρμογή τους σε συγκεκριμένη συνάρτηση.

- **Κεφάλαιο 3 - Βελτιστοποίηση**. Οι τεχνικές βελτιστοποιήσεως χρησιμοποιούνται για την ελαχιστοποίηση τιμών παραστάσεων. Ωστόσο και το σφάλμα προσεγγίσεως ενός **τεχνητού νευρωνικού δικτύου** μπορεί να θεωρηθεί ως μία παράσταση προς ελαχιστοποίηση. Στο κεφάλαιο αυτό εξετάζονται τεχνικές για την εύρεση τοπικών και καθολικών ελαχίστων και παρουσιάζεται το σύστημα βελτιστοποιήσεως MERLIN.

- **Κεφάλαιο 4 - Τεχνητά Νευρωνικά Δίκτυα**. Στο κεφάλαιο αυτό παρουσιάζονται τα προβλήματα που πραγματεύονται τα **τεχνητά νευρωνικά δίκτυα** και αναπτύσσονται διάφορες τοπολογίες τους.

- **Κεφάλαιο 5 - Παράλληλος Προγραμματισμός**. Στο κεφάλαιο αυτό αναπτύσσονται οι περισσότερες τεχνικές **παράλληλου προγραμματισμού**, πολλές από τις οποίες δοκιμάσθηκαν στα πρώιμα στάδια κατασκευής του μοντέλου και απορρίφθηκαν. Στο τέλος του κεφαλαίου αναπτύσσεται εκτενώς η μεθοδολογία **παράλληλου προγραμματισμού** που επιλέκτηκε για την ανάπτυξη του μοντέλου.

- **Κεφάλαιο 6 - Το Προτεινόμενο Μοντέλο**. Στο κεφάλαιο αυτό διατυπώνεται πλήρως το μοντέλο **προσαρμογής δεδομένων** που προτείνει αυτή η εργασία τόσο στην μία διάσταση όσο και στις δύο διαστάσεις. Σε αυτό το κεφάλαιο έχουμε την ένωση των συμπερασμάτων των τρίων προηγουμένων κεφαλαίων για την δημιουργία του προτεινομένου μοντέλου.

- **Κεφάλαιο 7 - Πειραματικά Αποτελέσματα**. Το προτεινόμενο μοντέλο δοκιμάσθηκε για την ακρίβεια των αποτελεσμάτων του σε μία σειρά από συναρτήσεις τόσο στην μία όσο και στις δύο διαστάσεις. Στο κεφάλαιο αυτό παρουσιάζονται αυτά τα αποτελέσματα και διατυπώνονται κάποια συμπεράσματα για την επιτυχία του μοντέλου.

- **Κεφάλαιο 7 - Μελλοντικές Επεκτάσεις**. Μετά την διατύπωση των συμπερασμάτων του μοντέλου στο κεφάλαιο αυτό παρουσιάζονται κάποιες επεκτάσεις που θα μπορούσαν να γίνουν στο προτεινόμενο μοντέλο ώστε να γίνει αποδοτικότερο αλλά και να μπορέσει να χρησιμοποιηθεί σε διαφορετικά γνωστικά πεδία από αυτά της **προσαρμογής δεδομένων**.

Κεφάλαιο 2

Παρεμβολή

2.1 Το πρόβλημα

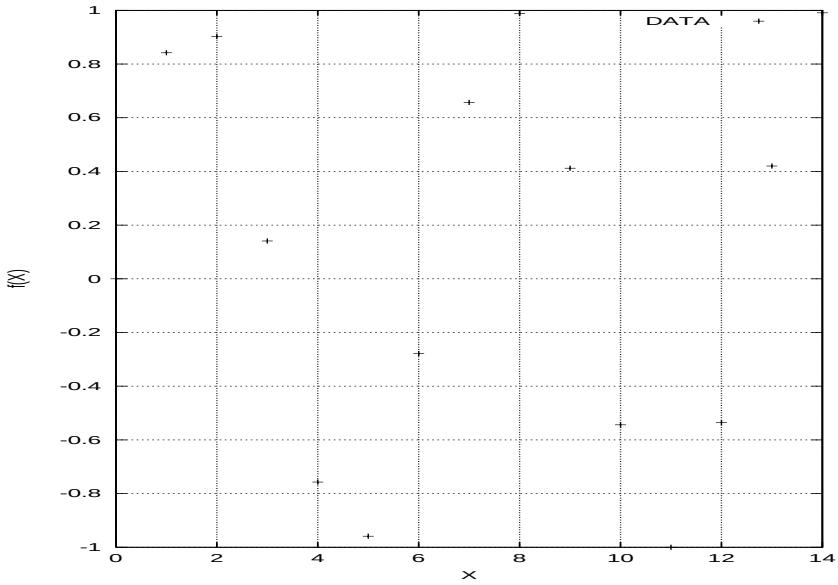
Το πρόβλημα που θα μας απασχολήσει στην συνέχεια αυτού του βιβλίου είναι η προσέγγιση συνεχών συναρτήσεων με την χρήση «μοντέλων». Αυτό που θα δίνεται σαν είσοδος στο πρόγραμμά μας είναι ένα σύνολο από σημεία της μορφής (x,f(x)) με το x να είναι ένα διάνυσμα στο $\Re^N$ και την συνάρτηση f να είναι μία απεικόνιση

$$f : \Re^N \rightarrow \Re$$

Έστω για παράδειγμα πως έχουμε τα δεδομένα του επόμενου πίνακα :

X	f(X)
0	0.0
1	0.842
2	0.903
3	0.141
4	-0.757
5	-0.959
6	-0.279
7	0.657
8	0.989
9	0.412
10	-0.544
11	-1.000
12	-0.536
13	0.420
14	0.991

Τα παραπάνω δεδομένα αν τα απεικονίσουμε στους άξονες X,Y θα πάρουμε το Σχήμα 2.1.



Σχήμα 2.1: Σημεία Μετρήσεως

Αυτό που έχουμε είναι σημεία. Αν θέλουμε να βρούμε τις πιθανές τιμές που έχει η συνάρτηση σε ενδιάμεσα σημεία μπορούμε να χρησιμοποιήσουμε κάποιο μοντέλο, που θα έχει τις ακόλουθες ιδιότητες:

1. Στα δοθέντα σημεία θα επιστρέφει την πειραματική τιμή.
2. Σε ενδιάμεσα σημεία θα επιστρέφει μία ικανοποιητική προσέγγιση της συναρτήσεως.

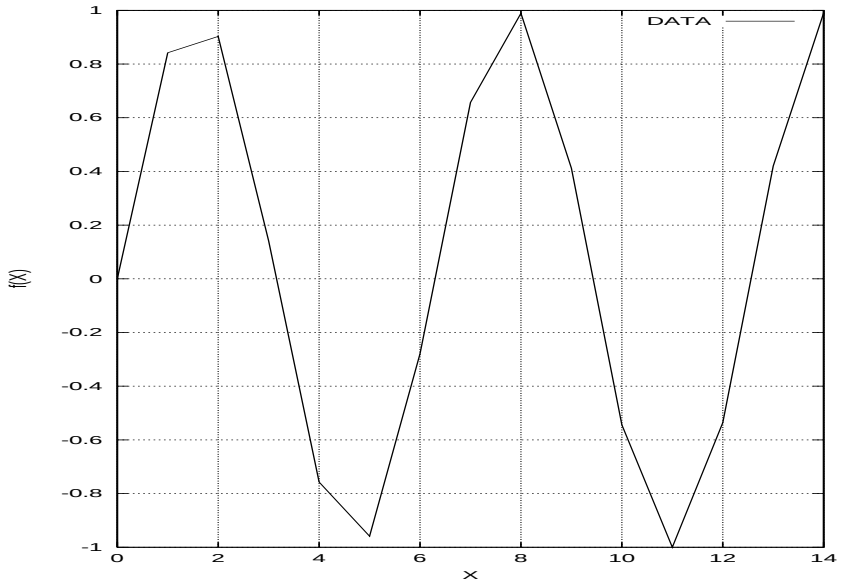
Αν για παράδειγμα ενώσουμε τα πειραματικά σημεία με ευθείες τότε το αποτέλεσμα που θα πάρουμε φαίνεται στο Σχήμα 2.2.

Η πραγματική συνάρτηση που πρέπει να προσεγγίσουμε είναι η $\sin x$ και η σύγκριση του παραπάνω απλού μοντέλου με την πραγματική συνάρτηση φαίνεται στο Σχήμα 2.3.

2.2 Ορισμός

Αν έχουμε N+1 σημεία[1] από μία συνάρτηση f άγνωστης μορφής θα τα λέμε σημεία παρεμβολής. Αυτό που ζητάμε είναι με κάποιον εύκολο τρόπο να προσεγγίσουμε τις τιμές αυτής της συναρτήσεως με την χρήση μίας συναρτήσεως γνωστής μορφής. Η συνήθης επιλογή για αυτές τις συναρτήσεις είναι τα πολυώνυμα, καθώς διαθέτουν μία σειρά από χρήσιμες ιδιότητες όπως:

[1] Ζεύγη της μορφής (X,Ψ)



Σχήμα 2.2: Προσέγγιση σημείων μετρήσεως με ευθείες

1. Υπολογίζονται πολύ εύκολα.
2. Παραγωγίζονται εύκολα.
3. Ολοκληρώνονται εύκολα.

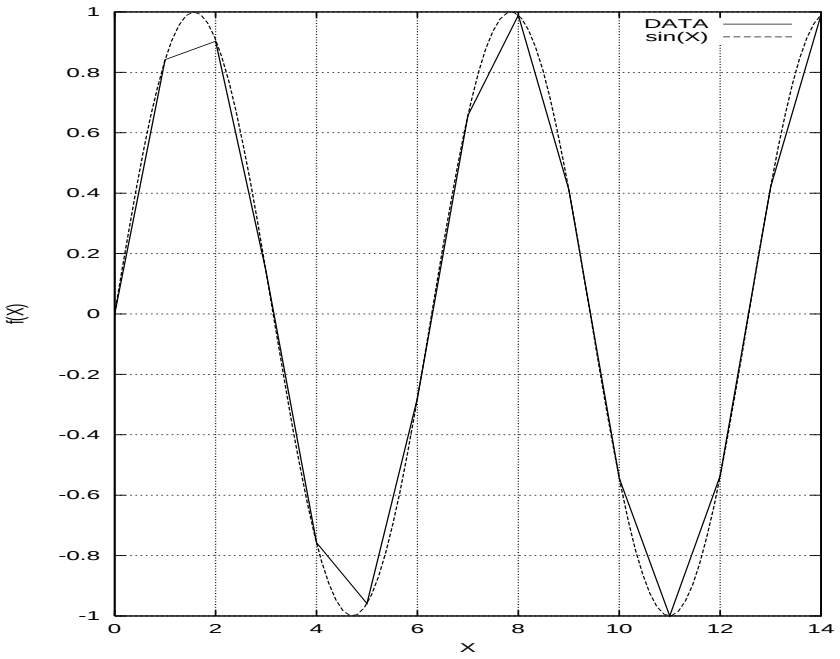
Θα θέλαμε με αυτά τα σημεία να δημιουργήσουμε ένα πολυώνυμο P του μικρότερου δυνατού βαθμού τέτοιο ώστε στα σημεία παρεμβολής να ισχύει

$$P(x_i) = f(x_i) = y_i$$

Υπάρχουν σοβαροί λόγοι για τους οποίους θέλουμε πολυώνυμα του μικρότερου δυνατού βαθμού. Καταρχήν όσο πιο μικρός είναι ο βαθμός ενός πολυωνύμου τόσο πιο γρήγορα αυτό υπολογίζεται. Κατά δεύτερον όπως θα δείξουμε στην συνέχεια υπάρχουν παθολογικές περιπτώσεις για τις οποίες η προσέγγιση με μεγάλου βαθμού πολυώνυμα αποτυγχάνει. Με άλλα λόγια ψάχνουμε να βρούμε απλά πολυώνυμα τα οποία να περιγράφουν όσο γίνεται καλύτερα την συνάρτησή μας.

2.3 Παρεμβολή Lagrange

Η πρώτη μορφή παρεμβολής στην οποία θα αναφερθούμε είναι η παρεμβολή Lagrange. Το πολυώνυμο παρεμβολής ορίζεται σύμφωνα με την εξίσωση:



Σχήμα 2.3: Σύγκριση πραγματικής συναρτήσεως συναρτήσεως - μοντέλου προσεγγίσεως με ευθείες

$$P(x) = \sum_{i=0}^{N} L_i(x) y_i \tag{2.1}$$

Αν θέλουμε να είμαστε σύμφωνοι με τις βασικές αρχές της παρεμβολής θα πρέπει στα σημεία παρεμβολής να ισχύει $P(x_i) = y_i$. Από την παραπάνω σχέση βλέπουμε πως ένας εύκολος τρόπος για να το πετύχουμε αυτό είναι ορίσουμε την συνάρτηση $L_i(x)$ με τον ακόλουθο τρόπο:

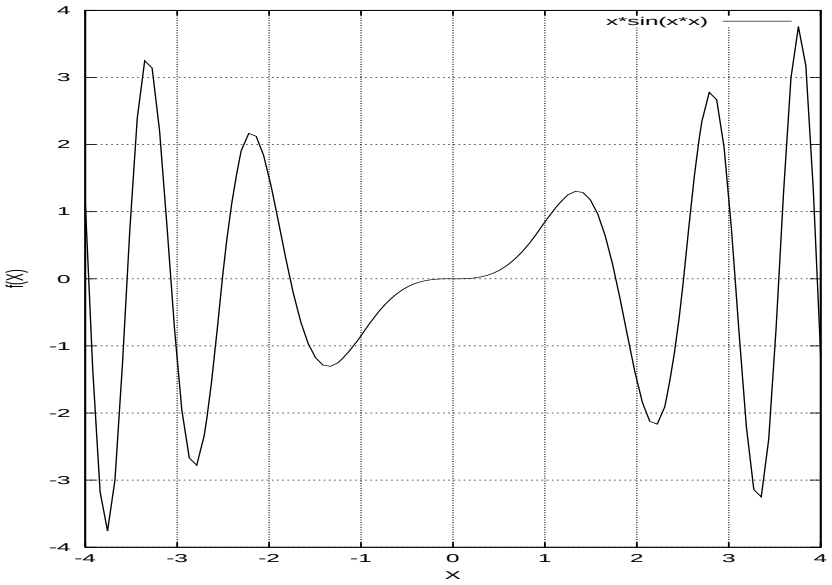
$$L_k(x_i) = \begin{cases} 1 & for \quad k = i \\ 0 & for \quad k \neq i \end{cases} \tag{2.2}$$

Η παραπάνω αναπαράσταση δεν είναι αρκετά βολική και επομένως θα θέλαμε έναν καλύτερο ορισμό όπως αυτός της εξισώσεως 2.3:

$$L_k(x) = \prod_{i=0, i\neq k}^{N} \frac{x - x_i}{x_k - x_i} \tag{2.3}$$

Αν προσέξουμε την παραπάνω σχέση θα δούμε πως γίνεται 1 μόνον στο επιθυμητό σημείο παρεμβολής και 0 σε κάθε άλλο σημείο παρεμβολής.

Το θέμα που τίθεται είναι κατά πόσον είναι καλή αυτή η μέθοδος παρεμβολής. Για να το εξακριβώσουμε αυτό θα θεωρήσουμε ένα παράδειγμα μίας συναρτήσεως



Σχήμα 2.4: Η συνάρτηση $x \sin x^2$ στο διάστημα [-4,4]

με δύσκολη μορφή που θα χρησιμοποιήσουμε πολλές φορές σε αυτό το κείμενο. Η συνάρτηση αυτή είναι η $x \sin x^2$ της οποίας η γραφική παράσταση στο διάστημα [-4,4] είναι στο Σχήμα 2.4.

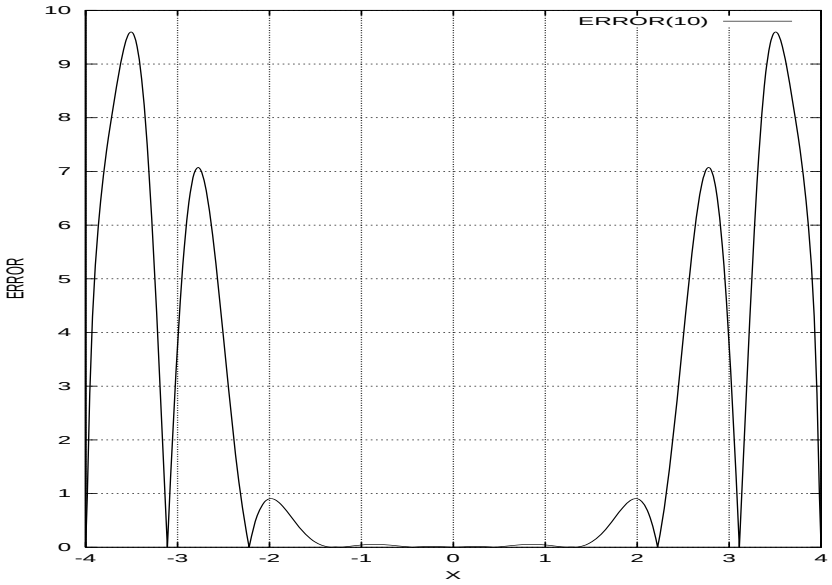
Πρώτα θα δοκιμάσουμε να προσεγγίσουμε την παραπάνω συνάρτηση με 10 σημεία παρεμβολής. Το σφάλμα προσεγγίσεως δίνεται από το Σχήμα 2.5, που στον οριζόντιο άξονα έχουμε τα πειραματικά σημεία και στον κατακόρυφο άξονα το αντίστοιχο λάθος.

Στα άκρα του πεδίου ορισμού η συνάρτηση παρουσιάζει και την μεγαλύτερη δυσκολία να προσεγγιστεί. Κάτι τέτοιο πρέπει να το περιμένουμε αν κοιτάξουμε προσεκτικά την δομή που έχει η γραφική παράσταση της συναρτήσεως. Για να εξακριβώσουμε ποιος είναι ο «κατάλληλος» βαθμός του πολυωνύμου προσεγγίσεως μπορούμε να δοκιμάσουμε να προσεγγίσουμε τα δεδομένα μας και με εικοστού βαθμού πολυώνυμα Lagrange. Σε αυτήν την περίπτωση το σφάλμα προσεγγίσεως της παρεμβολής Lagrange είναι στο Σχήμα 2.6.

Αν και το σφάλμα προσεγγίσεως είναι κοντά στο μηδέν στο μεγαλύτερο μέρος του δείγματος στα άκρα το σφάλμα είναι αρκετά μεγαλύτερο από όταν χρησιμοποιήσαμε 10 όρους στο πολυώνυμο Lagrange. Αντί λοιπόν το συνολικό σφάλμα να μειωθεί αυξήθηκε κατά πολύ. Αυτό το πρόβλημα της παρεμβολής το είχαν παρατηρήσει οι μαθηματικοί και για άλλες συναρτήσεις όπως η συνάρτηση του Runge με τύπο

$$f(x) = \frac{1}{1 + 25x^2} \tag{2.4}$$

Αν μπορέσουμε να αλλάξουμε λίγο την διαμέριση βάζοντας περισσότερα σημεία στα άκρα τότε θα έχουμε επιτύχει τον στόχο μας. Ένας αρκετά επιτυχημένος



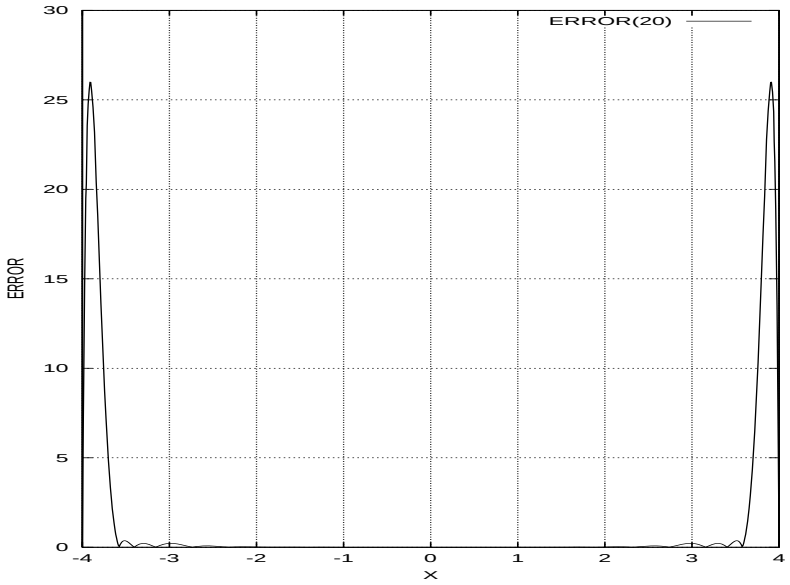
Σχήμα 2.5: Το σφάλμα προσεγγίσεως της παρεμβολής Lagrange με 10 όρους για την συνάρτηση $x \sin x^2$

τρόπος διαμερίσεως που έχει προταθεί είναι αυτός των πολυωνύμων Chebysev που ορίζονται ως εξής[5]:

$$c_i = \cos \frac{\pi(2i+1)}{2(N+1)} \tag{2.5}$$

Αυτός ο τρόπος διαμερίσεως δίνει σημεία στο [-1,1] αλλά πολλαπλασιάζοντάς τον με το πλάτος της συναρτήσεως που έχουμε (στην περίπτωσή μας 4) μπορούμε να το φέρουμε στο επιθυμητό διάστημα. Χρησιμοποιώντας 100 σημεία και τον τρόπο που δίνει η Εξίσωση (2.5) καταλήγουμε στο διάγραμμα σφάλματους του Σχήματος 2.7.

Αν εξαιρέσουμε το μεγάλο σφάλμα στα πρώτα σημεία του διαστήματος γενικά η προσέγγιση ήταν αρκετά καλύτερη από τις προηγούμενες, αλλά σε καμία περίπτωση δεν μπορούμε να θεωρήσουμε πως έχουμε λύσει το πρόβλημα. Από την άλλη χρησιμοποιήσαμε πολυώνυμα μεγάλου βαθμού για να επιτύχουμε την παραπάνω προσέγγιση, κάτι που κοστίζει σε μνήμη και σε χρόνο. Όπως θα δούμε στην συνέχεια αυτής της εργασίας θα ψάξουμε να βρούμε προσεγγίσεις σχετικά φτηνές σε μνήμη και σε χρόνο που πάνω από όλα να γενικεύονται σε κάθε πρόβλημα.



Σχήμα 2.6: Το σφάλμα προσεγγίσεως της παρεμβολής Lagrange με 20 όρους για την συνάρτηση $x \sin x^2$

2.4 Κυβικές Πολυωνυμικές Splines

2.4.1 Διαμέριση

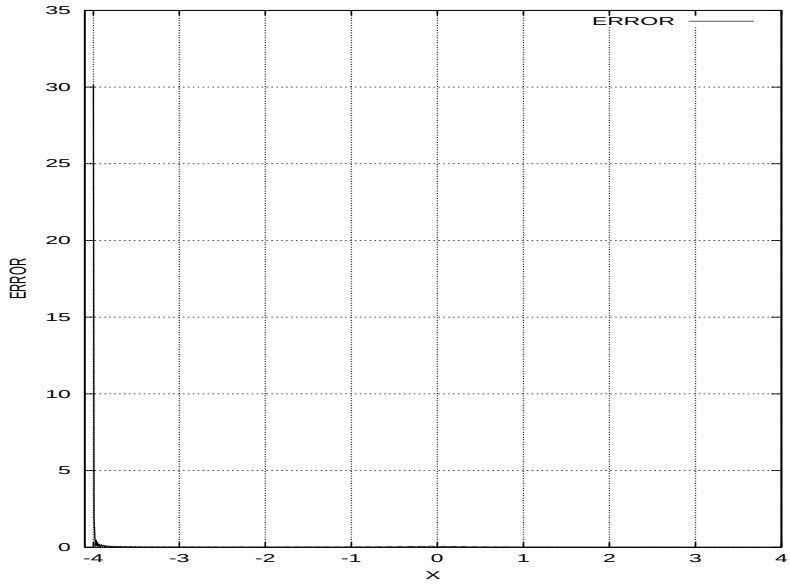
Στην παρεμβολή που είδαμε μέχρι στιγμής διαπιστώσαμε πως είναι σχετικά καλή λύση σε μικρές περιοχές. Αυτό σημαίνει πως αν διαμερίσουμε ένα μεγάλο διάστημα σε πολλά μικρότερα θα επιτύχουμε (πιθανόν) καλά αποτελέσματα βάζοντας σε καθένα διάστημα ένα καλό μοντέλο. Αν λοιπόν έχουμε το διάστημα $[a, b]$ και το χωρίσουμε σε N διαστήματα χρησιμοποιώντας την διαμέριση:

$$a = x_0 < x_1 < ... < x_N = b$$

τότε κάθε υποδιάστημα θα ορίζεται από τα σημεία $[x_i, x_{i+1}]$. Σε καθένα τέτοιο υποδιάστημα μπορούμε να χρησιμοποιήσουμε κάποιο μοντέλο όπως πολυώνυμα ενός συγκεκριμένου βαθμού. Αυτές οι συναρτήσεις προσεγγίσεως λέγονται splines. Οι splines που χρησιμοποιούνται περισσότερο είναι πολυώνυμα.

2.4.2 Παρεμβολή με κυβικά φυσικά splines

Οι κυβικές splines είναι πολυώνυμα τρίτου βαθμού. Για να δημιουργήσουμε τέτοιες splines αρκεί να ορίσουμε σε κάθε διάστημα ένα πολυώνυμο τρίτου βαθμού και στην συνέχεια να απαιτήσουμε να υπάρχει συνέχεια τιμής, πρώτης παραγώγου και δεύτερης παραγώγου στα σημεία παρεμβολής. Για να γίνει αυτό η επίλυση ενός συστήματος εξισώσεων gauss του οποίου η τελική μορφή θα καθοριστεί από



Σχήμα 2.7: Το σφάλμα της παρεμβολής Lagrange με 100 όρους και διαμέριση Chebysev για την συνάρτηση $x \sin x^2$

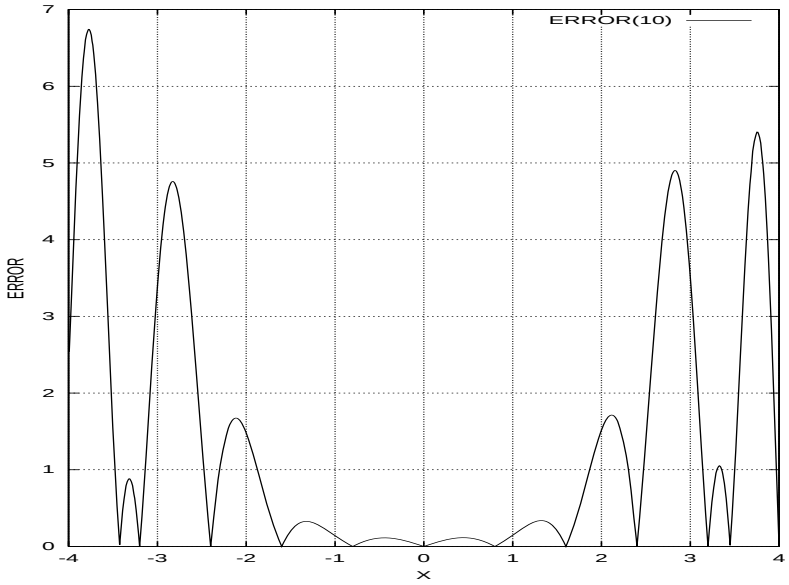
τις υποθέσεις που κάνουμε τόσο για την μορφή του πολυωνύμου παρεμβολής όσο και για τις συνθήκες που θα ισχύουν για την δεύτερη παράγωγο αυτού. Στα παραδείγματα αυτού του κεφαλαίου χρησιμοποιήθηκαν οι κυβικές φυσικές splines των οποίων ο μαθηματικός ορισμός δίνεται ως ακολούθως[1]:

$$\begin{gathered} s_{i+1}(x_i) = f(x_i) \\ s_i(x_i) = f(x_i) \\ s_i'(x_i) = s_{i+1}'(x_i) \\ s_i''(x_i) = s_{i+1}''(x_i) \\ s''(x_0) = 0 \\ s''(x_N) = 0 \end{gathered} \tag{2.6}$$

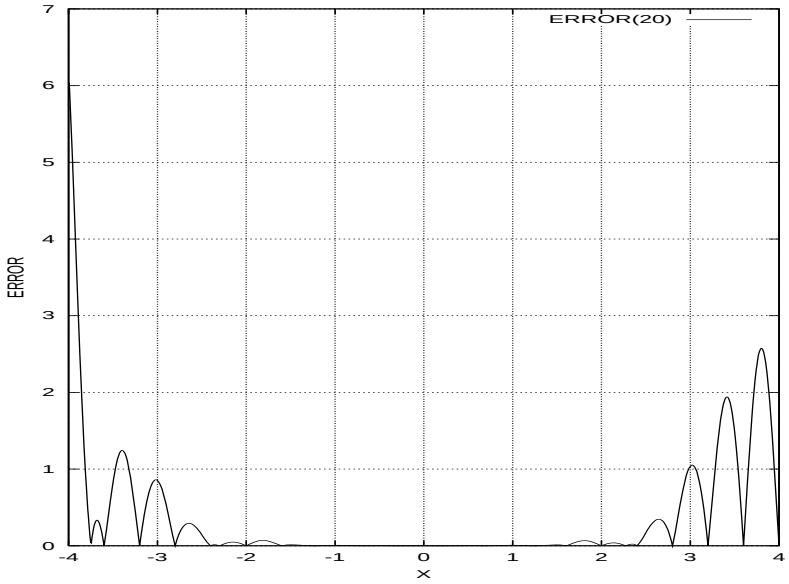
Αν θέλουμε να δοκιμάσουμε την επίδοση της παραπάνω μεθοδολογίας δεν έχουμε παρά να χρησιμοποιήσουμε το πρόβλημα της $x \sin x^2$. Για 10 διαστήματα το σφάλμα προσεγγίσεως δίνεται στο Σχήμα 2.8.

Αν χρησιμοποιήσουμε 20 διαστήματα στην διαμέριση θα λάβουμε το λάθος που δίνεται στο Σχήμα 2.9

Στην μέση του γραφήματος που η συνάρτηση $x \sin x^2$ μοιάζει λίγο με την x^3 η προσέγγιση ήταν αρκετά ικανοποιητική. Αλλά στα άκρα που η συνάρτηση είναι περισσότερο μη ομαλή το αποτέλεσμα στην προσέγγιση δεν μπορεί να θεωρηθεί ικανοποιητικό. Το παραπάνω πείραμα έγινε και με 50 διαστήματα. Εκεί η προσέγγιση ήταν καλύτερη, αλλά εξακολουθούσαν να υπάρχουν μεγάλα λάθη στα άκρα.



Σχήμα 2.8: Το σφάλμα προσεγγίσεως με κυβικές φυσικές splines 10 διαστημάτων για την συνάρτηση $x \sin x^2$



Σχήμα 2.9: Το σφάλμα προσεγγίσεως με κυβικές φυσικές Splines 20 διαστημάτων για την συνάρτηση $x \sin x^2$

Κεφάλαιο 3

Βελτιστοποίηση

3.1 Βασικοί ορισμοί

Για να δώσουμε έναν σαφή ορισμό της βελτιστοποιήσεως θα πρέπει σε πρώτη φάση να εξετάσουμε ποια είναι τα προβλήματα που πραγματεύεται η βελτιστοποίηση. Οι κατηγορίες αυτών των προβλημάτων είναι:

1. Τοπική βελτιστοποίηση.
2. Καθολική βελτιστοποίηση.

3.2 Τοπική βελτιστοποίηση

3.2.1 Βελτιστοποίηση χωρίς περιορισμούς

Η βελτιστοποίηση χωρίς περιορισμούς είναι η περίπτωση που θα μας απασχολήσει και περισσότερο στην συνέχεια αυτού του κειμένου. Γενικά αυτό που θέλουμε σε αυτήν την περίπτωση είναι ένα σημείο στο N-διάστατο χώρο για το οποίο σημείο έχουμε την ελάχιστη τιμή κάποιας συναρτήσεως. Με την χρήση μαθηματικου συμβολισμού αυτό μπορεί να εκφραστεί ως ακολούθως:

$$x_* = argmin_x(f(x)) \tag{3.1}$$

δηλαδή η $f(x_*)$ είναι μικρότερη ή ίση της $f(x)$ για κάθε x που είναι στην γειτονιά του x_*.

3.2.2 Βελτιστοποίηση με περιορισμούς

Στην βελτιστοποίηση με περιορισμούς γενικά μπορούμε να θεωρήσουμε πως έχουμε δύο συναρτήσεις που χρησιμοποιούμε: Την συνάρτηση προς ελαχιστοποίηση με το όνομα f και την συνάρτηση των περιορισμών με το όνομα c. Στην γενική περίπτωση εφαρμογής μεθόδων επιλύσεως μπορούμε να θεωρήσουμε πως και οι δύο

συναρτήσεις είναι μη γραμμικές. Δύο πολύ συνηθισμένοι τύποι βελτιστοποιήσεως με περιορισμούς είναι οι ακόλουθοι [**??**]:

$$min\{f(x) :\ c_i(x) \leq 0, i \in I_1, c_i(x) = 0, i \in I_2\} \tag{3.2}$$

$$min\{f(x) : c(x) = 0, l \leq x \leq u\} \tag{3.3}$$

Στην πρώτη περίπτωση περιορισμών χωρίζουμε το πλήθος των περιορισμών σε δύο σύνολα με τα ονόματα I_1 και I_2. Το πρώτο σύνολο είναι εκείνοι οι περιορισμοί που είναι αρνητικοί για κάθε διάνυσμα x που έχουν ως είσοδο και το δεύτερο είναι εκείνο το σύνολο των περιορισμών που μηδενίζονται για κάθε είσοδο. Ένα παράδειγμα αυτών των περιορισμών είναι το ακόλουθο:

$$\begin{gathered} f(x_1, x_2) = x_1^2 + 3x_2^2 \\ x_1 x_2 \leq 0 \\ x_1^3 - 1 = 0 \end{gathered}$$

Στην δεύτερη περίπτωση περιορισμών οι περιορισμοί είναι μηδενικοί όταν το διάνυσμα εισόδου είναι σε κάποιο αποδεκτό διάστημα τιμών. Δηλαδή φράσουμε τις τιμές των διανυσμάτων εισόδου ανάμεσα σε δύο όρια. Όπως θα δούμε αργότερα στις μεθόδους βελτιστοποιήσεως κάτι τέτοιο είναι απολύτως θεμιτό προκειμένου να μην λαμβάνουν εξαιρετικά μεγάλες απόλυτες τιμές τα σημεία συναρτήσεως που θα μας οδηγούν σε λάθη υπερχειλίσεως.

Κλείνοντας αυτήν την υποενότητα θα πρέπει να τονιστεί πως οι περιορισμοί ισχύουν καθόλη την διάρκεια της μεθόδου της βελτιστοποιήσεως.

3.2.3 Μη γραμμικά ελάχιστα τετράγωνα

Όταν η συνάρτηση που θέλουμε να ελαχιστοποιήσουμε εκφράζεται[1] με την μορφή αθροίσματος τετραγώνων τότε μιλάμε για βελτιστοποίηση ελαχίστων τετραγώνων. Φυσικά από την στιγμή που έχουμε άθροισμα τετραγώνων δεν μπορεί η συνάρτηση που θέλουμε να αποτιμήσουμε να πάρει αρνητικές τιμές. Γενικά μπορούμε να εκφράσουμε το πρόβλημα ως ακολούθως [**??**]:

$$min\left\{r(x) : r(x) = \sum_{i=1}^{N} f_i(x)^2\ x \in \Re^N\right\} \tag{3.4}$$

3.3 Μέθοδοι βελτιστοποιήσεως

Σε αυτήν την ενότητα θα ασχοληθούμε με τις διάφορες μεθόδους βελτιστοποιήσεως που συναντάμε στα προβλήματά μας. Όπως θα δούμε και στην συνέχεια αυτής της αναφοράς πολλές από αυτές τις μεθόδους είναι κρίσιμες για την πορεία της βελτιστοποιήσεως τόσο από απόψεως αποδόσεως αλλά και ως προς το σημείο χρήσεώς τους.

[1] Η μπορεί να εκφραστεί

3.3.1 Γενική μέθοδος βελτιστοποιήσεως

Αν θέλουμε να περιγράψουμε μία γενική μέθοδο βελτιστοποιήσεως αυτή θα μπορούσε να συμπτυχθεί σε τρία βήματα[2]:

1. Εύρεση της κατευθύνσεως στην αναζήτηση $s^{(k)}$.
2. Ελαχιστοποίηση $f(x^{(k)} + \lambda s^{(k)})$ ως προς λ. Αυτό το βήμα ονομάζεται γραμμική αναζήτηση.
3. $x^{(k+1)} = x^{(k)} + \lambda s^{(k)}$

3.3.2 Η μέθοδος των εναλλασσομένων διευθύνσεων

Αν θεωρήσουμε πως τα στοιχεία του διανύσματος των μεταβλητών είναι ασυσχέτιστα μεταξύ τους τότε μπορούμε να εφαρμόσουμε την μέθοδο των εναλλασσομένων διευθύνσεων που γενικά μπορεί να θεωρηθεί αρκετά γρήγορη. Η βασική αρχή της μεθόδου είναι πως προσπαθεί να κάνει βελτιστοποίηση ως προς καθένα από τα στοιχεία του διανύσματος μεταβλητών σε επαναλαμβανόμενους κύκλους μέχρι συγκλήσεως. Φυσικά όπως είναι προφανές αυτή η προσέγγιση δεν είναι ικανοποιητική, αφού δεν μπορούμε να υποθέτουμε κάθε φορά πως οι μεταβλητές σε μία συνάρτηση είναι ασυσχέτιστες μεταξύ τους. Μία αρκετά διαδομένη παραλλαγή αυτής της μεθοδολογίας είναι η ROLL τεχνική, που βηματικά μπορεί να περιγραφεί ως ακολούθως [6]:

Αλγόριθμος Roll

Είσοδος:Η συνάρτηση f
Έξοδος:Το διάνυσμα παραμέτρων x.

1. Για κάθε μεταβλητή x_i
   - (α) θεωρούμε πως υπάρχει μία κατεύθυνση s_i.
   - (β) $f_+ = f(x_1, .., x_i + s_i, .., x_N)$
   - (γ) Αν $f_+ \leq f(x)$ τότε $s_i = a * s_i,\ a > 0$
   - (δ) Αλλιώς
     - i. $f_- = f(x_1, ..., x_i - s_i, ..., x_N)$
     - ii. Αν $f_- \leq f(x)$ τότε $s_i = -a * s_i$
     - iii. Αλλιώς $s_i = -\frac{1}{2} * \frac{f_+ - f_-}{f_+ + f_- - 2*f} * s_i$
2. Γραμμική αναζήτηση στην κατεύθυνση $(s_1, s_2, ..., s_N)$.

[2]Υποθέτουμε πως εκτελείται το k βήμα

3.3.3 Η μέθοδος του Newton για ελαχιστοποίηση

Για να προσεγγίσουμε την συνάρτησή μας χρησιμοποιούμε ανάπτυγμα Taylor δεύτερης τάξεως, οπότε και θα έχουμε την Εξίσωση (3.5)[6]:

$$f(x+h) \cong f(x) + \bigtriangledown f(x)^T * h + \frac{1}{2} * h^T * \bigtriangledown^2 f(x) * h \tag{3.5}$$

Για ευκολία θα χρησιμοποιήσουμε τους συμβολισμούς:

$$B = \bigtriangledown^2 f(x) \tag{3.6}$$

$$g = \bigtriangledown f(x) \tag{3.7}$$

Για να φτάσουμε σε ελάχιστη τιμή θα πρέπει να ισχύει

$$\bigtriangledown^2 f(x) * h = - \bigtriangledown f(x) \Rightarrow h = -B^{-1} * g \tag{3.8}$$

Το h ονομάζεται βήμα Newton. Επομένως καταφέραμε να ανάγουμε ένα σχετικά δύσκολο πρόβλημα σε επίλυση ενός γραμμικού συστήματος. Η παραπάνω μεθοδολογία είναι ικανοποιητική όταν βρισκόμαστε σχετικά κοντά στην λύση και ο πίνακας B είναι θετικά ορισμένος. Σε αντίθετη περίπτωση θα πρέπει να χρησιμοποιήσουμε κάποια τροποποίηση της μεθόδου του Newton για βελτιστοποίηση.

3.3.4 Περιοχή Εμπιστοσύνης

Ένας καλός τρόπος να ψάξουμε στην γειτονιά της λύσεως είναι και η χρήση τεχνικών περιοχής εμπιστοσύνης. Ουσιαστικά είναι πρόβλημα βελτιστοποιήσεως υπό περιορισμούς. Η συνάρτηση προς ελαχιστοποίηση είναι η [7]:

$$f(x^{(k)} + h) \simeq q(h) = f(x^{(k)}) + h^T * g^{(k)} + \frac{1}{2} * h^T * B * h \tag{3.9}$$

Με δεδομένο πως $||h|| < R$. Η περιοχή R ονομάζεται και περιοχή εμπιστοσύνης. Ο όρος $g^{(k)}$ που βλέπουμε να μετέχει στην έκφραση ορίζεται ως $g^{(k)} = \bigtriangledown f(x^{(k)})$. Με την χρήση των μεθόδων περιοχής εμπιστοσύνης και με τους πολλαπλασιαστές Lagrange μπορούμε να οδηγηθούμε σε έναν τρόπο επιλύσεως που αποτελεί παραλλαγή της μεθόδου Newton:

$$(B^{(k)} + \mu * I) * h^{(k)} = -g^{(k)},\ ||h^{(k)}(\mu)|| \leq R \tag{3.10}$$

Αυτό που άλλαξε είναι ότι προστέθηκε ο ανισοτικός περιορισμός για το βήμα h καθώς και ένας αριθμός μ που κατευθύνει και την πορεία της βελτιστοποιήσεως.

3.3.5 Η μέθοδος Levenberg

Μία μέθοδος που είναι κατάλληλη για ελάχιστα τετράγωνα είναι η μέθοδος Levenberg. Η μέθοδος αυτή[3] χρησιμοποιεί μεθόδους περιοχής εμπιστοσύνης. Η

[3] Έχει υλοποιηθεί αρχικά στο πακέτο MINPACK

αντικειμενική συνάρτηση μπορεί να γραφεί $f(x) = \sum_{i=1}^{M} f_i^2(x) \equiv r(x)^T * r(x)$: όπου ισχύει $r(x)^T = (f_1(x), f_2(x), ..., f_M(x))^T$. Σε αυτήν την περίπτωση ορίζουμε J τον Ιακωβιανό πίνακα $J_{ij} = \frac{\partial f_i(x)}{\partial x_j}$ και D είναι ένας διαγώνιος πίνακας. Η τετραγωνική προσέγγιση στο (x+h) δίνεται από τον τύπο $f(x+h) \simeq f(x) + g^T(x) * h + \frac{1}{2} * h^T * B(x) * h$. Η προηγούμενη έκφραση ελαχιστοποιείται υπό την συνθήκη $\|D * h\| \leq R$, με R να είναι η ακτίνα της περιοχής εμπιστοσύνης. Ο αλγόριθμος μπορεί σε γενικές γραμμές να περιγραφεί ως εξής [7]:

Αλγόριθμος Levenberg

Είσοδος:Η συνάρτηση f, η ακτίνα εμπιστοσύνης R.
Έξοδος:Το διάνυσμα παραμέτρων x.

1. Για i=1..N κάνε $D_{ii}^0 = \left\| \frac{\partial r(x^{(0)})}{\partial x_i} \right\|$
2. k=1.
3. Αν $\left\| D^{(k)} * h^{(k)}(0) \right\| \leq R^{(k)}$ τότε
   (α) $\delta^{(k)} = h^{(k)}(0)$
4. Αλλιώς
   (α) Εύρεση $\lambda^{(k)} > 0$ ώστε $\left\| D^{(k)} * h^{(k)}(\lambda^{(k)}) \right\| = R^{(k)}$
   (β) $\delta^{(k)} = h^{(k)}(\lambda^{(k)})$
5. Αν $f(x^{(k)} + \delta^{(k)}) < f(x^{(k)})$ τότε
   (α) $x^{(k+1)} = x^{(k)} + \delta^{(k)}$
   (β) Υπολογισμός $J^{(k+1)}$.
6. Αλλιώς
   (α) $x^{(k+1)} = x^{(k)}$.
   (β) $J^{(k+1)} = J^{(k)}$.
7. Υπολογισμός νέου $R^{(k)}$ με την χρήση της τεχνικής DOG-LEG.
8. $D_{ii}^{(k+1)} = \max D_{ii}^{(k)}, \left\| \frac{\partial r(x^{(k+1)})}{\partial x_i} \right\| \forall i = 1, 2, .., N$
9. k=k+1.
10. GOTO 3

3.3.6 Η μέθοδος DFP

Ο Εσσιανός πίνακας B δεν είναι εύκολο να υπολογιστεί, καθώς απαιτεί δύο παραγωγίσεις στην σειρά και επιπλέον ο αποθηκευτικός χώρος είναι τετραγωνικός. Επομένως αυτό που μπορούμε να κάνουμε είναι να προσεγγίζουμε τον Εσσιανό πίνακα και τον αντίστροφό του με κάποια ευρετική τεχνική. Με την χρήση της μεθόδου DFP [6] μπορούμε να υπολογίσουμε την προσέγγιση $B^{(k+1)}$ για τον Εσσιανό πίνακα ή την προσέγγιση $H^{(k+1)}$ για τον αντίστροφο του Εσσιανού, θεωρώντας πως εκτελείται το βήμα k, σύμφωνα με τις εξισώσεις (3.11) και (3.12):

$$B^{(k+1)} = \left(I - \frac{dh^T}{d^T h}\right) B^{(k)} \left(I - \frac{dh^T}{d^T h}\right) + \frac{dd^T}{d^T h} \tag{3.11}$$

$$H^{(k+1)} = H^{(k)} + \frac{hh^T}{d^T h} - \frac{H^{(k)} dd^T H^{(k)}}{d^T H^{(k)} d} \tag{3.12}$$

Στις εξισώσεις αυτές θεωρούμε πως $d = g^{(k+1)} - g^{(k)}$ και $h = x^{(k+1)} - x^{(k)}$.

3.3.7 Η μέθοδος BFGS

Μία διαφορετική μέθοδος για τον υπολογισμό του Εσσιανού πίνακα και του αντίστροφου αυτού είναι η BFGS[6]. Ο υπολογισμός των δύο προσεγγίσεων δίνεται στις εξισώσεις

$$B^{(k+1)} = B^{(k)} + \frac{dd^T}{d^T h} - \frac{B^{(k)} hh^T B^{(k)}}{h^T B^{(k)} h} \tag{3.13}$$

$$H^{(k+1)} = \left(I - \frac{hd^T}{d^T h}\right) H^{(k)} \left(I - \frac{hd^T}{d^T h}\right) + \frac{hh^T}{d^T h} \tag{3.14}$$

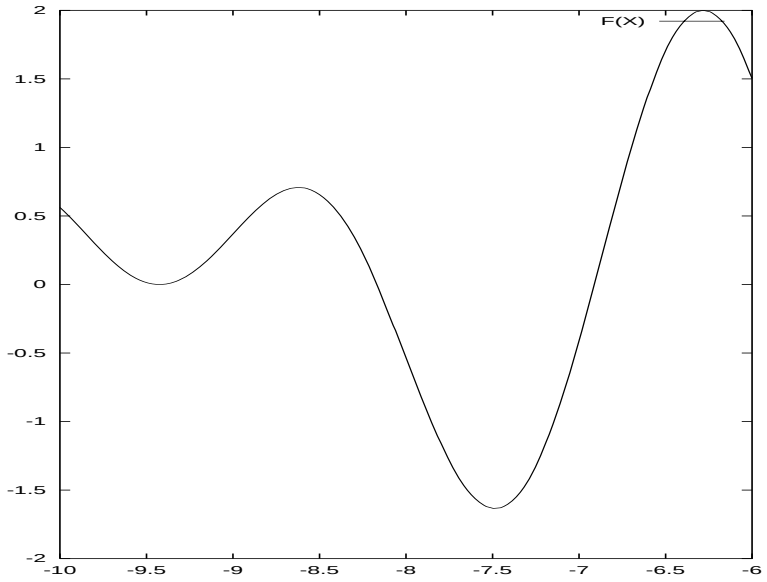
3.4 Ολική Βελτιστοποίηση

Οι μέθοδοι βελτιστοποιήσεως που συζητήθηκαν προηγουμένως δεν επιτυγχάνουν πάντοτε το καλύτερο δυνατό αποτέλεσμα. Έστω για παράδειγμα την ελαχιστοποίηση μιας συναρτήσεως με την μορφή που εμφανίζεται στο Σχήμα 3.1

Το «καθολικό» ελάχιστο της συναρτήσεως είναι στο σημείο -7.5 Το σημείο -9.5 επίσης ικανοποιεί συνθήκες αριστότητας και άρα είναι πιθανόν να είναι το αποτέλεσμα μίας μεθόδου τοπικής ελαχιστοποιήσεως. Όπως θα δούμε στην συνέχεια υπάρχουν μέθοδοι που προσπαθούν να εντοπίσουν το καθολικό ελάχιστο «ξεφεύγοντας» από τοπικά ελάχιστα.

3.4.1 Random Restarts

Είναι η απλούστερη τεχνική καθολικής βελτιστοποιήσεως. Με αυτήν την απλή μέθοδο για πολλές τυχαίες αρχικοποιήσεις των παραμέτρων του μοντέλου μας βρίσκουμε κάποιες τιμές για το λάθος προσεγγίσεως. Η καλύτερη τιμή από αυτές που βρίσκουμε είναι και αυτή που θα κρατήσουμε. Αν θέλουμε να δούμε αλγοριθμικά αυτήν την διαδικασία θα έχουμε



Σχήμα 3.1: Καθολικά ελάχιστα

Αλγόριθμος Random

Είσοδος: Τα δεδομένα Δ που θέλουμε να προσεγγίσουμε και ένας αριθμός επαναλήψεων N.
Έξοδος: Το διάνυσμα Π των παραμέτρων του μοντέλου και η τιμή της συναρτήσεως E.

1. i=0
2. $\Pi_i = Random()$.
3. $E_i = F(\Pi_i, \Delta)$.
4. i=i+1
5. if(i<N) goto 2
6. $\Pi^* = argmin_{\Pi_i}(F(\Pi_i, \Delta))$
7. $E^* = F(\Pi^*, \Delta)$

Η παραπάνω τεχνική δεν εγγυάται την εύρεση κάποιας λύσεως καθώς σε κάθε βήμα της ανακυκλώσεως ξεχνάει τα ελάχιστα και αρχίζει και πάλι από την αρχή. Έτσι μπορεί να καταλήξει πολλές φορές στο ίδιο ελάχιστο.

3.4.2 Clustering

Είναι η μέθοδος καθολικής βελτιστοποιήσεως που θα χρησιμοποιήσουμε στα πειράματα του κεφαλαίου 6. Η βασική ιδέα αυτών των μεθόδων είναι πως ξεκινούμε από ένα αρχικό πληθυσμό ομοιόμορφα κατανεμημένων σημείων από τα οποία δημιουργούμε ομάδες. Τελικός σκοπός είναι να χρησιμοποιούμε μόνον μία φορά μία μέθοδο τοπικής ελαχιστοποιήσεως σε κάθε τέτοια ομάδα. Μία παραλλαγή του αλγορίθμου CLUSTERING είναι η SINGLE LINKAGE CLUSTERING, την οποία παραθέτουμε αλγοριθμικά παρακάτω[4]:

Αλγόριθμος CLUSTERING

Είσοδος: N ο αριθμός σημείων του δείγματος, γ το ποσοστό των σημείων που δεν απορρίπτονται
Έξοδος: X^*τα τοπικά ελάχιστα που βρίσκονται.

1. Επέλεξε N τυχαία σημεία για αρχικό δείγμα.
2. Επιλογή των γN σημείων με την χαμηλότερη τιμή.
3. Για καθένα από τα γN εκτέλεσε μια τοπική ελαχιστοποίηση ενός βήματος.
4. Ομαδοποίηση των σημείων.
5. SET1=όλα τα σημεία που δεν έχουν μπει σε ομάδα.
6. Αν $|Set1| = 0$ GOTO 11.
7. $x^{(1)}$= το σημείο με την χαμηλότερη τιμή στο SET1.
8. x^*=ΤΟΠΙΚΗ_ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ($x^{(1)}$)
9. Αν $x^* \notin X^*$τότε
   (α) $X^* = X^* \cup \{x^*\}$
   (β) GOTO 4.
10. Αλλιώς GOTO 1.
11. Τέλος και επιστροφή του σημείου με την χαμηλότερη τιμή στο X^* ως καθολικό ελάχιστο.

3.5 Merlin - Mcl

Το πακέτο Merlin/Mcl[3,7] είναι ένα πακέτο λογισμικού που χρησιμοποιείται για πολυδιάστατη βελτιστοποίηση. Merlin είναι το βασικό σύστημα με τους υλοποιη-

μένους αλγορίθμους και Mcl[4] είναι μία γλώσσα που χρησιμοποιεί το Merlin και μπορούμε με πιο εύκολο τρόπο να κατευθύνουμε την βελτιστοποίηση. Το πακέτο ξεκίνησε να υλοποιείται στα μέσα της δεκαετίας του 80 και σήμερα βρίσκεται στην έκδοση 3.0.3 Το πακέτο μπορεί να υλοποιήσει πολυδιάστατη ελαχιστοποίηση με ανισωτικούς περιορισμούς για κάθε μεταβλητή.

Για την αντιμετώπιση των προβλημάτων ελαχιστοποιήσεως παρέχεται ένα εύχρηστο σύστημα διαπροσωπείας με τον χρήστη με αρκετά διαφωτιστικό σύστημα βοήθειας που μπορεί ανά πάσα στιγμή να εμφανιστεί. Επίσης θα πρέπει να σημειωθεί πως το σύστημα είναι ανοικτό σε επεκτάσεις και αυτό μπορεί να μας διευκολύνει όταν ζητάμε να προσθέσουμε κάποια νέα τεχνική που δεν είναι υλοποιημένη στο σύστημα.

3.5.1 Υλοποιημένοι αλγόριθμοι

Στο σύστημα έχουν υλοποιηθεί αλγόριθμοι με χρήση παραγώγων αλλά και μέθοδοι που δεν χρησιμοποιούν παραγώγους. Εννοείται πως η δεύτερη ομάδα τεχνικών έχει ταχύτερη σύγκλιση από την πρώτη, αλλά απαιτείται μεγαλύτερη εργασία από τον χρήστη. Οι μέθοδοι που έχουν υλοποιηθεί χωρίς την χρήση παραγώγων είναι:

1. Simplex.
2. Roll. Είναι μέθοδος ιχνοαναζητήσεως. Η περιγραφή της έχει δοθεί προηγουμένως.

Οι μέθοδοι που χρησιμοποιούν παραγώγους είναι οι ακόλουθες:

1. Μέθοδοι συζυγών κλίσεων.
2. Μεταβλητής μετρικής (BFGS, DFP).
3. Levenberg - Marquadrt.

3.5.2 Απαιτήσεις από τον χρήστη

Ο χρήστης θα πρέπει να φτιάξει σε Fortran77 την αντικειμενική συνάρτηση. Αν θέλει μπορεί να γράψει την κλίση της συναρτήσεως και τον Εσσιανό πίνακα, αλλά αν δεν το κάνει το Merlin μπορεί να τα υπολογίσει με διαδοχικές κλήσεις της αντικειμενικής συναρτήσεως.

3.5.3 Παραδείγματα χρήσεως

Στα επόμενα παρατίθενται μερικά παραδείγματα χρήσεως του συστήματος λογισμικού Merlin, τόσο με απλή βελτιστοποίηση όσο και με ελάχιστα τετράγωνα.

[4] Merlin Control Language

3.5.3.1 Παράδειγμα Funmin

Ας θεωρήσουμε το παράδειγμα $f(x_1, x_2) = x_1^2 + (x_2 - 1)^4$. Η κλίση αυτής της συναρτήσεως ως προς κάθε μεταβλητή είναι $\frac{\partial f(x)}{\partial x_1} = 2 * x_1$ και $\frac{\partial f(x)}{\partial x_2} = 4(x_2 - 1)^3$. Ο Εσσιανός πίνακας μπορεί να γραφεί ως ακολούθως:

$$\begin{bmatrix} 2 & 0 \\ 0 & 12(x_2 - 1)^2 \end{bmatrix}$$

3.5.3.2 Παράδειγμα Subsum

Το προηγούμενο παράδειγμα μπορεί να γραφεί και σε μορφή αθροίσματος τετραγώνων. Σε αυτήν την περίπτωση η συνάρτηση γράφεται ως: $f(x) = \sum_{i=1}^{2}(f_i^2(x))$ με $f_1(x) = x1$ και $f_2(x) = (x_2 - 1)^2$. Αν θέλουμε στην περίπτωση που έχουμε άθροισμα τετραγώνων μπορούμε να χρησιμοποιήσουμε και την Ιακωβιανή συνάρτηση που ορίζεται ως

$$J_{ij}(f) = \frac{\partial f_i(x)}{\partial x_j} \tag{3.15}$$

Η Ιακωβιανή για το δεδομένο πρόβλημα είναι ο πίνακας:

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 * (x_2 - 1) \end{bmatrix}$$

Κεφάλαιο 4

Νευρωνικά Δίκτυα

4.1 Νευρωνικά ενός επιπέδου

4.1.1 Διαχωρισμός δύο ομάδων

Αν υποθέσουμε πως διαθέτουμε δύο ομάδες δεδομένων, τότε μία απλή συνάρτηση διακρίσεως θα μπορούσε να είναι η εξής:

$$y(x) = w^T x + w_0 \tag{4.1}$$

Όπου το διάνυσμα w ονομάζεται βάρος και η παράμετρος w_0 ονομάζεται πόλωση ή ακόμα και threshold. Αυτή η συνάρτηση μπορεί να θεωρηθεί η καλύτερη για την διάκριση ενός επιπέδου σε δύο ομάδες: αυτές που για τα σημεία τους ισχύει $y(x) < 0$ και αυτές για τις οποίες έχουμε $y(x) > 0$. Είναι δηλαδή σαν να έχουμε την εξίσωση μίας ευθείας και αρκεί να υπολογίσουμε τις παραμέτρους της προκειμένου να επιτύχουμε την καλύτερη διάκριση. Αν θέλουμε να εκφράσουμε και αριθμητικά την παραπάνω εξίσωση μπορούμε να την γράψουμε σαν[8]:

$$y(x) = \sum_{i=1}^{k} (x_i w_i) + w_0 \tag{4.2}$$

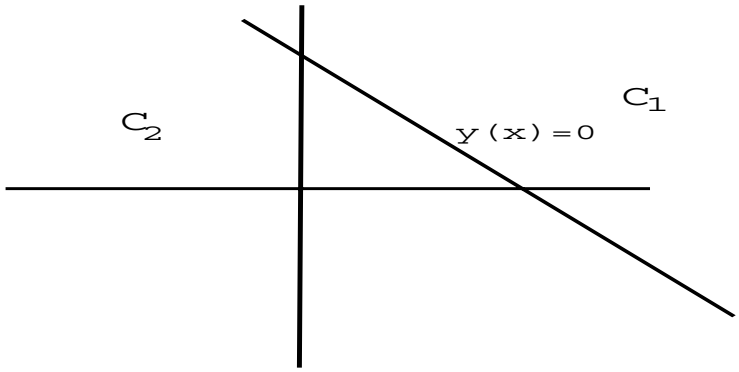
Σαν k θεωρούμε την διάσταση των προτύπων εισόδου. Η προηγούμενη εξίσωση μπορεί να γραφεί και με πιο συμπαγή τρόπο, ως ακολούθως:

$$y(x) = \sum_{i=0}^{k} \widetilde{x_i} w_i \tag{4.3}$$

Υπο την παραδοχή πως για το νέο διάνυσμα $\widetilde{x}$ ισχύει:

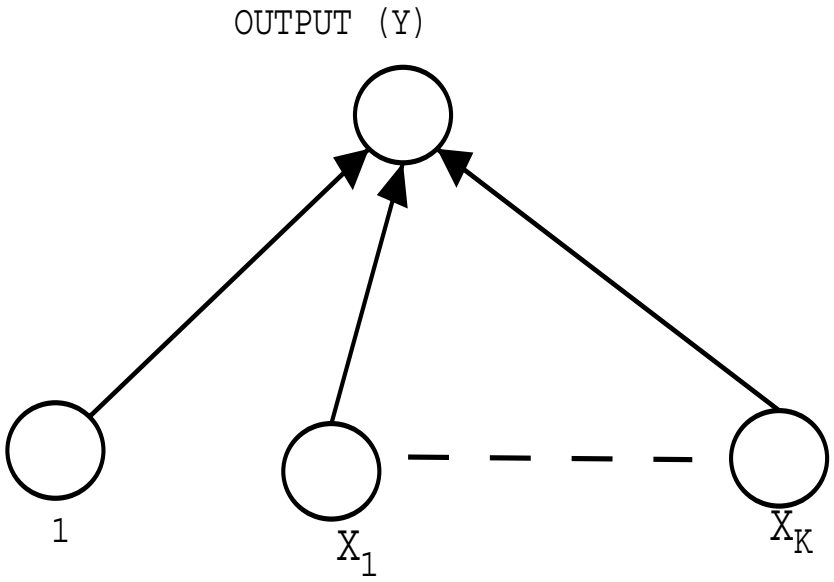
$$\widetilde{x_i} = \begin{cases} 1, i = 0 \\ x_{i,} otherwise \end{cases} \tag{4.4}$$

Για την περίπτωση που έχουμε διάσταση προτύπων 2 τότε μπορούμε γραφικά να παραστήσουμε τον γραμμικό διαχωριστή με Σχήμα 4.1:



Σχήμα 4.1: Διαχωρισμός κατηγοριών στις δύο διαστάσεις με την χρήση ευθείας

Όπου με C1, C2 συμβολίσαμε τις δύο ομάδες που υποθέτουμε πως έχει ένας χώρος. Φυσικά για να αποφευχθούν αποτυχίες κατηγοριοποιήσεως θα πρέπει κοντά στην ευθεία διαχωρισμού να έχουμε αρκετά σημεία, καθώς εκεί είναι που μεγαλώνει ο κίνδυνος να κάνουμε λάθος. Αν θέλουμε να παραστήσουμε γραφικά έναν διαχωριστή σαν και τον παραπάνω θα πάρουμε ένα σχήμα που μοιάζει αρκετά με την ιδεατή δομή ενός δικτύου:



Σχήμα 4.2: Perceptron ενός επιπέδου

4.1.2 Διαχωρισμός πολλών ομάδων

Αν δεν έχουμε μόνον δύο τάξεις και έχουμε περισσότερες τότε το μοντέλο που περιγράψαμε στην προηγούμενη υποενότητα θα πρέπει να επεκταθεί, ώστε να καλύψει και αυτήν την περίπτωση. Για αυτόν τον λόγο θα πρέπει να χρησιμοποιήσουμε κάποιες ιδέες και τεχνικές στις οποίες έχουμε ήδη αναφερθεί. Αυτές οι τεχνικές έχουν να κάνουν με τις συναρτήσεις διακρίσεως. Στις συναρτήσεις διακρίσεως επιτυγχάνουμε διάκριση όταν η συνάρτηση λαμβάνει την μεγαλύτερη τιμή της. Έτσι ο τύπος 4.1 μπορεί να αλλάξει λίγο στην Εξίσωση (4.5):

$$y_i(x) = w_{T\,i}x + w_{i0} \tag{4.5}$$

Όπου το i συμβολίζει τον αριθμό της τάξεως για την οποία ενδιαφερόμαστε κάθε φορά.

4.1.3 Προσέγγιση Συναρτήσεων

Στον τομέα των νευρωνικών δικτύων μία πολύ σημαντική εργασία είναι η προσέγγιση συναρτήσεων. Το πρόβλημα αυτό ορίζεται ως εξής:

Δεδομένων κάποιων σημείων (x_i, y_i) προσαρμόσετε σε αυτά ένα νευρωνικό δίκτυο N(x,p) που αναπαριστά τα δεδομένα μας ικανοποιητικά. Το διάνυσμα p είναι οι παράμετροι του νευρωνικού δικτύου.

4.1.4 Δίκτυα Perceptron

Τα πιο διαδεδομένα δίκτυα ενός επιπέδου είναι τα δίκτυα Perceptron. Αυτό που αξίζει να δούμε εδώ είναι η εκπαίδευση Perceptron και το θεώρημα συγκλήσεως Perceptron.

4.1.4.1 Εκπαίδευση Perceptron

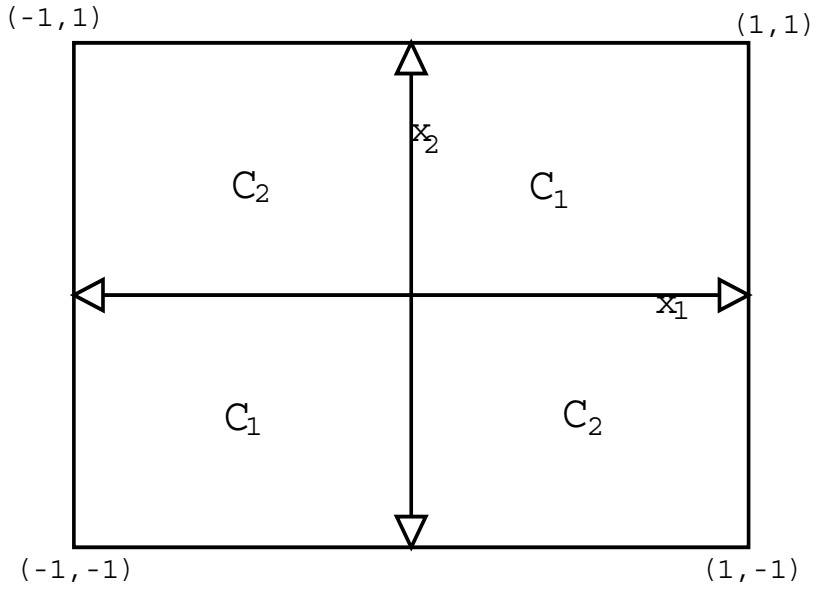
Βηματικά η εκπαίδευση Perceptron μπορεί να οριστεί ως ακολούθως[8]:

Αλγόριθμος Perceptron

Είσοδος:Τα σημεία $(x, t(x))$ του συνόλου εκπαιδεύσεως και ο ρυθμός μαθήσεως n.
Έξοδος:Το διάνυσμα παραμέτρων w.

1. Αρχικοποίηση των βαρών στο διάστημα [-1,1].
2. Για κάθε σημείο x
   (α) Αποτιμάται η τιμή της εξόδου $y(x)$ για αυτό το σημείο.
   (β) Αν $y(x)t(x) < 0$ τότε $w_i^{(t+1)} = w_i^{(t)} + n(t(x) - y(x))$, $i = 1..k$, όπου $0 < n < 1$ ο ρυθμός μαθήσεως.
3. Αν όλα τα σημεία ταξινομήθηκαν σωστά τότε ΔΙΑΚΟΠΗ.
4. Αλλιώς μετάβαση στο 2.

Για κάποιες περιπτώσεις ο παραπάνω αλγόριθμος μπορεί και να μην συγκλίνει. Χαρακτηριστικό είναι το παράδειγμα της συναρτήσεως XOR στο οποίο διαχωρίζουμε τον χώρο σύμφωνα με το Σχήμα 4.3



Σχήμα 4.3: Το πρόβλημα του XOR στις δύο διαστάσεις
για το οποίο δεν υπάρχει σύγκληση με τον αλγόριθμο Perceptron.

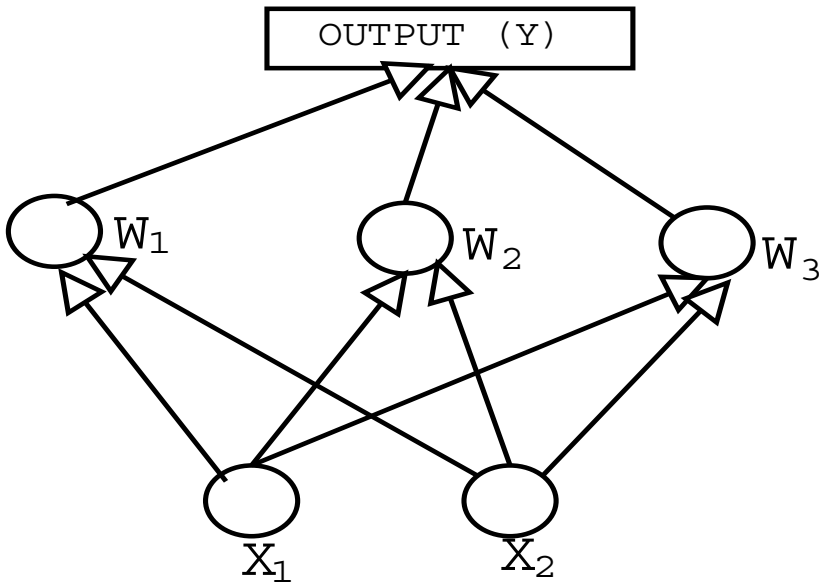
4.1.4.2 Θεώρημα συγκλήσεως Perceptron

Το θεώρημα συγκλίσεως Perceptron αναφέρει πως αν τα πρότυπα δύο κατηγοριών είναι γραμμικώς διαχωρίσιμα τότε σε πεπερασμένο πλήθος βημάτων με τον αλγόριθμο Perceptron θα καταλήξουμε στην λύση.

4.2 Πολυεπίπεδα νευρωνικά δίκτυα

4.2.1 Τοπολογίες Δικτύων

Αν δεν θέλουμε να έχουμε τους περιορισμούς των μονοεπίπεδων νευρωνικών δικτύων τότε μπορούμε να περάσουμε σε πιο πολύπλοκα, ως προς την δομή τους, νευρωνικά δίκτυα. Στα πολυεπίπεδα νευρωνικά δίκτυα η τελική έξοδος δεν είναι απλώς ένα σταθμισμένο άθροισμα των εισόδων, αλλά θεωρούμε πως μπορεί να περάσουμε την είσοδό μας από πολλά επίπεδα με διαφορετικά βάρη σε κάθε περίπτωση. Κάθε επίπεδο ανάλογα με την θέση του στο νοητό γράφημα απεικονίσεως του δικτύου παίρνει και διαφορετικό όνομα. Έτσι το πρώτο επίπεδο έχει επικρατήσει να το ονομάζουμε **επίπεδο εισόδου** και το τελικό επίπεδο **επίπεδο εξόδου**. Τα ενδιάμεσα επίπεδα χαρακτηρίζονται με τους όρους **κρυμμένα επίπεδα**, **επίπεδα επεξεργασίας** κτλ. Σε αυτό το κείμενο θα χρησιμοποιούμε τον όρο κρυμμένο επίπεδο. Μία πολύ απλή τοπολογία για τα πολυεπίπεδα νευρωνικά δίκτυα φαίνεται στο Σχήμα 4.4. Στο σχήμα περιγράφεται ένα νευρωνικό δίκτυο με δύο εισόδους, τρεις μονάδες στο κρυμμένο επίπεδο και μία μονάδα στο επίπεδο εξόδου:



Σχήμα 4.4: Πολυεπίπεδο Perceptron

Τα βέλη στο παραπάνω γράφημα περιγράφουν ροή της πληροφορίας από κόμβο σε κόμβο. Φυσικά δεν είναι απαραίτητο να συνδέονται όλοι οι κόμβοι με όλους. Επίσης μπορεί να έχουμε σύνδεση από επίπεδο σε μεθεπόμενα επίπεδα. Στην εργασά θα ασχοληθούμε με νευρωνικά δίκτυα που ταιριάζουν με την παραπάνω τοπολογία, καθώς είναι η πιο συνηθισμένη μορφή στην προσέγγιση συναρτήσεων.

4.2.2 Έξοδος κόμβων

Σε ένα πολυεπίπεδο νευρωνικό δίκτυο κάθε μονάδα δέχεται κάποιον αριθμό εισόδων και παράγει κάποια έξοδο. Αυτή η έξοδος μεταβιβάζεται σε κόμβους στο επόμενο επίπεδο συνδέσεως, πιθανόν μετά από στάθμιση. Γενικά μπορούμε να θεωρήσουμε πως σε ένα νευρωνικό δίκτυο feed forward η έξοδος ενός κόμβου μπορεί να δοθεί από τον τύπο[8]:

$$z_k = g\left(\sum_i w_{ki} * z_i\right) \tag{4.6}$$

Με τον όρο w_{ki} συμβολίζουμε το βάρος της συνδέσεως από τον κόμβο i του προηγούμενου επιπέδου με τον κόμβο k του παρόντος επιπέδου. Με τον όρο z_i συμβολίζουμε την έξοδο της μονάδας i. Πρέπει να προσεχτεί πως όταν είμαστε στο επίπεδο εισόδου τότε οι έξοδοι των μονάδων αυτών είναι φυσικά τα στοιχεία των προτύπων όπως αυτά εισάγονται. Η συνάρτηση g ονομάζεται συνάρτηση ενεργοποιήσεως και μπορεί να έχει διάφορες παραστάσεις όπως:

1. Συνάρτηση threshold με τύπο

$$g(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{4.7}$$

2. Συνάρτηση υπερβολικής εφαπτομένης.

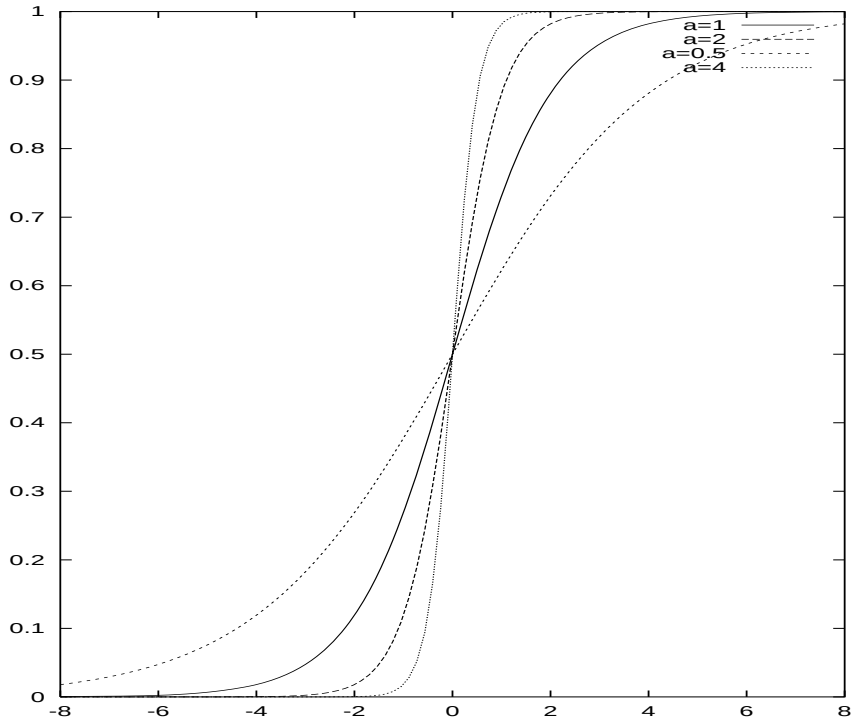
3. Σιγμοειδής συνάρτηση.

4.2.3 Σιγμοειδείς μονάδες

Μία συνάρτηση που χρησιμοποιείται πάρα πολύ στα νευρωνικά δίκτυα και στα οικονομικά μοντέλα είναι η σιγμοειδής συνάρτηση. Ο γενικός της τύπος είναι ο ακόλουθος:

$$\sigma(x) = \frac{1}{1 + exp(-ax)} \tag{4.8}$$

με το $a > 0$. Για διάφορες τιμές του a έχουμε τις ακόλουθες γραφικές παραστάσεις:



Σχήμα 4.5: Σιγμοειδείς συναρτήσεις για διαφορετικές τιμές του a

Όλες οι σιγμοειδείς προς τα δεξιά πάνε στο 1 και προς τα αριστερά στο 0. Όσο πιο μεγάλος γίνεται ο συντελεστής a τόσο πιο πολύ μοιάζει το γράφημα με την συνάρτηση threshold, αλλά η σιγμοειδής έχει το βασικό πλεονέκτημα απέναντί της: είναι συνεχώς παραγωγίσιμη.

4.2.4 Η αρχή της παγκόσμιας προσεγγίσεως

Η αρχή της παγκόσμιας προσεγγίσεως αναφέρει[11]:

Ένα νευρωνικό δίκτυο με ένα κρυμμένο επίπεδο και με επαρκή αριθμό κόμβων μπορεί να προσεγγίσει οποιαδήποτε συνάρτηση.

Φυσικά λέγοντας επαρκή αριθμό αυτός μπορεί να είναι αρκετά μεγάλος.

4.2.5 Back Propagation

Άσχετα με το αν έχουμε διαχωρισμό σε κλάσεις ή προσέγγιση συναρτήσεων πάντα θα υπάρχει κάποιο σφάλμα, καθώς δεν είναι απαραίτητο πως θα πετύχουμε από την αρχή σύκλιση του μοντέλου μας. Γενικά έχει επικρατήσει στα νευρωνικά δίκτυα να εκφράζουμε αυτό το λάθος με την χρήση του τετραγωνικού σφάλματος

ή πιο φορμαλιστικά:

$$E = \frac{1}{2}\sum_{i=1}^{N}(y_i - t_i)^2 \tag{4.9}$$

Όπου με y_i συμβολίζουμε την έξοδο του νευρωνικού δικτύου και με t_i την επιθυμητή έξοδο, όπως μας έχει δοθεί κατά την εκπαίδευση του συστήματος. Όπως είχαμε πει και στο κεφάλαιο της βελτιστοποιήσεως ένας καλός δρόμος στην ελαχιστοποίηση μίας συναρτήσεως βρίσκεται με παραγώγιση της συναρτήσεως. Έτσι και στα νευρωνικά δίκτυα πρέπει να βρούμε την παράγωγο του λάθους ως προς τις παραμέτρους του δικτύου. Φυσικά όταν έχουμε να κάνουμε με συνάρτηση τόσο περίπλοκη όπως οι συναρτήσεις λάθους ενός νευρωνικού δικτύου, τότε χρειαζόμαστε κάποιον συστηματικό τρόπο για τον υπολογισμό των παραγώγων. Αυτός ο τρόπος ονομάζεται Back Propagation. Για τον υπολογισμό των παραγώγων χρησιμοποιούμε τους όρους δέλτα που ορίζονται ως εξής:

$$\delta_k = \begin{cases} y_k - t_k, \, output\, layer \\ z_k(1 - z_k)\sum_{j=1}^{c} w_{jk}\delta_j, \, otherwise \end{cases} \tag{4.10}$$

όπου k είναι ο όρος για τον οποίο θέλουμε να υπολογίσουμε την παράγωγο. Όπως βλέπουμε έχουμε προώθηση του λάθους από κόμβους που είναι στο επόμενο επίπεδο προς τους κόμβους που είναι σε προηγούμενο επίπεδο. Βέβαια από την στιγμή που έχουμε τους όρους δέλτα πρέπει να βρούμε τις παραγώγους του λάθους ως προς τις παραμέτρους του δικτύου:

$$\frac{\partial E}{\partial w_{kj}} = \delta_k z_j \tag{4.11}$$

4.3 Gradient Descent

Μία αρκετά διαδεδομένη τεχνική βελτιστοποιήσεως των νευρωνικών δικτύων είναι η μέθοδος gradient descent. Μπορούμε να διακρίνουμε αυτήν την μέθοδο σε δύο παραλλαγές: την online εκπαίδευση και την offline. Στην πρώτη περίπτωση τα πρότυπα εμφανίζονται με την σειρά στο δίκτυο και κάθε πρότυπο ενημερώνει τα βάρη. Στην δεύτερη περίπτωση αφού περάσουν όλα τα πρότυπα ενημερώνουμε τα βάρη. Διάφορες μελέτες έχουν δείξει πως η πρώτη μέθοδος είναι ισοδύναμη και καλύτερη της δεύτερης. Επίσης έχει λιγότερες απαιτήσεις, αφού δεν είναι απαραίτητο πάντα τα πρότυπα που χρειαζόμαστε θα είναι διαθέσιμα από την αρχή. Στην περίπτωση που έχουμε online τεχνική ο κανόνας ενημερώσεως των βαρών δίνεται ως εξής:

$$\Delta w_{ji} = -n\frac{\partial E^k}{\partial w_{ji}} \tag{4.12}$$

Όπου με E^n συμβολίζουμε το σφάλμα για το πρότυπο n. Αν έχουμε offline τεχνική τότε ο κανόνας ενημερώσεως των βαρών αλλάζει σε:

$$\Delta w_{ji} = -n\frac{\partial E}{\partial w_{ji}} \tag{4.13}$$

4.4 Γενίκευση

Ακόμα και αν έχουμε επιτύχει σε ένα νευρωνικό δίκτυο να έχουμε πολύ καλό λάθος, αυτό δεν σημαίνει πως το νευρωνικό δίκτυο είναι χρήσιμο. Τα νευρωνικά δίκτυα δημιουργούνται για να δουλέψουν σε κάποιο άγνωστο περιβάλλον και όχι για να μάθουν κάποια δεδομένα. Η μάθηση δεδομένων θα μπορούσε να επιτευχθεί και με την επίλυση ενός γραμμικού συστήματος, το οποίο θα έβρισκε ακριβώς ποιες παράμετροι ταιριάζουν στα δεδομένα μας, με λιγότερο επίπονο τρόπο από ότι η εκπαίδευση ενός δικτύου. Επομένως θα πρέπει αφού βρούμε κάποιες καλές παραμέτρους για το νευρωνικό μας δίκτυο να δοκιμάσουμε την κατασκευή μας σε δεδομένα που πιθανόν δεν είχαν χρησιμοποιηθεί κατά την διάρκεια της εκπαιδεύσεως. Αυτό το σημείο ονομάζεται γενίκευση.

Κεφάλαιο 5

Παράλληλος Προγραμματισμός

5.1 Νήματα

Αν σε ένα πρόγραμμα θέλουμε με κάποιον τρόπο να εκτελούμε δύο εργασίες παράλληλα, αυτό σε καμία περίπτωση δεν μπορεί να γίνει με κάποιον συμβατικό τρόπο. Αυτό που μπορούμε να κάνουμε είναι να χρησιμοποιήσουμε νήματα. Τα νήματα είναι ξεχωριστές διεργασίες οι οποίες έχουν τον ίδιο χώρο μνήμης μεταξύ τους. Έτσι μπορούν να έχουν πρόσβαση στα ίδια δεδομένα χωρίς να καταφεύγουμε σε λύσεις διαδιεργασιακής επικοινωνίας όπως η διαμοιραζόμενη μνήμη και οι sockets. Πάνω στα νήματα έχουν αναπτυχθεί αρκετές βιβλιοθήκες προγραμματισμού και σε αυτήν την ενότητα θα εξετάσουμε τις σημαντικότερες από αυτές.

5.1.1 Locks

Όταν έχουμε νήματα το βασικό τους πλεονέκτημα και ταυτόχρονα μειονέκτημα είναι πως όλες οι διεργασίες αναφέρονται στα ίδια δεδομένα. Αν λοιπόν «ταυτόχρονα» δύο διεργασίες επιχειρήσουν πρόσβαση στα ίδια δεδομένα το αποτέλεσμα θα είναι απρόβλεπτο. Για αυτόν τον λόγο χρησιμοποιούμε τα Locks. Είναι ειδικοί μηχανισμοί οι οποίοι συνήθως είναι σε μία από ένα σύνολο δίτιμων καταστάσεων: Ανοικτό και Κλειστό. Όταν είναι σε κατάσταση Κλειστό η διεργασία που επιχειρήσει πρόσβαση σε αυτά παγώνει μέχρι η διεργασία που τα έθεσε σε αυτήν την κατάσταση να τα επαναθέσει σε κατάσταση Ανοικτό. Αν λοιπόν κάθε διεργασία χρησιμοποιεί αυτούς του μηχανισμούς τότε μπορούμε να εγγυηθούμε κάποιου είδους ακεραιότητα για τα δεδομένα μας. Ο ακόλουθος ψευδοκώδικας δείχνει πως πρέπει να χρησιμοποιούνται:

- Lock(Mutex).
- Access(Data).

- Unlock(Mutex).

Φυσικά ο παραπάνω κώδικας θα πρέπει να υπάρχει σε κάθε διεργασία.

5.1.2 BB_THREADS

Είναι η πιο απλή υλοποίηση της πολυνηματικής επεξεργασίας στο Linux. Ο δικτυακός τους τόπος είναι το ftp://caliban.physics.utoronto.ca/pub/linux. Τα βασικά βήματα για την εκτέλεση ενός προγράμματος σε BB THREADS είναι τα εξής[9]:

1. Εκκίνηση του προγράμματος σαν μία διεργασία.
2. Εκτίμηση του μέγιστου μεγέθους δεδομένων στοίβας για κάθε νήμα (˜64K).
3. Αρχικοποίηση αριθμού Locks.
4. Προσθήκη κάθε νήματος.
5. Αναμονή μέχρι να τελειώσουν όλα τα νήματα.

Ένα βασικό μειονέκτημα που έχει η παραπάνω βιβλιοθήκη είναι η χρήση και διαχείριση των locks. Θα πρέπει από την αρχή να καθορίσουμε τον αριθμό των locks και η πρόσβαση σε αυτούς επιτυγχάνεται με την χρήση αριθμών, κάτι που δεν είναι αρκετά ευέλικτο.

5.1.3 Posix Threads

Είναι η πλέον διαδεδομένη βιβλιοθήκη νημάτων καθώς την συναντάμε σε αρκετά λειτουργικά συστήματα, αν και έχουν διαδοθεί περισσότερο σε συστήματα Linux παρά σε άλλα συστήματα τύπου UNIX, αφού εκεί έχουμε υλοποίηση των νημάτων από τον πηρύνα του λειτουργικού. Ο δικτυακός τους τόπος βρίσκεται στο site http://pauillac.inria.fr/˜xleroy/linuxthreads/ αν και συμπεριλαμβάνονται σε όλες τις σύγχρονες διανομές Linux και FreeBSD. Τα βασικά βήματα χρήσεώς τους είναι τα ακόλουθα:

1. Εκκίνηση του προγράμματος εφαρμογής.
2. Αρχικοποίηση Locks. Σε αντίθεση με την bb_threads, εδώ οι κλειδαριές είναι μεταβλητές και όχι αριθμοί, κάτι που μας επιτρέπει να εκτελούμε και πράξεις επί αυτών.
3. Εκκίνηση κάθε νήματος και πέρασμα παραμέτρων σε αυτά.
4. Αναμονή μέχρι τέλους των νημάτων.

5.2 Διαμοιραζόμενη μνήμη

Ένας ακόμη ενδιαφέρων τρόπος επικοινωνίας μεταξύ ταυτόχρονων διεργασιών είναι η διαμοιραζόμενη μνήμη. Με την διαμοιραζόμενη μνήμη[1] μπορούμε να έχουμε πολλές διεργασίες που να έχουν πρόσβαση στην ίδια μνήμη. Φυσικά προκειμένου να αποφευχθούν ασυνέπειες θα πρέπει να υπάρχουν μηχανισμοί κλειδώματος των κοινών περιοχών μνήμης. Τέτοιοι μηχανισμοί είναι οι σημαφόροι. Θα πρέπει να σημειωθεί πως η κοινή μνήμη δεν έχει σε τίποτα να κάνει με βιβλιοθήκες χρήστη, όπως τα νήματα, και έτσι μπορούν να βρεθούν στα περισσότερα σύγχρονα λειτουργικά συστήματα.

5.3 Memory Map

Άλλος ένας τρόπος διαδιεργασιακής επικοινωνίας είναι η απεικόνιση αρχείων στην μνήμη. Σύμφωνα με αυτήν την τεχνική τμήματα αρχείων απεικονίζονται σε συνεχόμενη περιοχή της μνήμης. Αρχικά αυτός ο τρόπος είχε χρησιμοποιηθεί για την γρήγορη επεξεργασία αρχείων, αλλά επεκτάθηκε για την διαδιεργασιακή επικοινωνία. Το σύστημα γραφικής διαπροσωπείας με τον χρήστη X βασίζει την ταχύτητα επικοινωνίας σε τέτοιες τεχνικές. Από την άλλη η πολύ χρήσιμη τεχνική των διαμοιραζόμενων βιβλιοθηκών βασίζεται αποκλειστικά σε αυτήν την τεχνική. Παρόλες τις παραπάνω πολύ χρήσιμες εφαρμογές, η τεχνική αυτή είναι υποδεέστερη σε ταχύτητα από την κοινή μνήμη και έτσι δεν θα αναλυθεί περισσότερο.

5.4 Clustering

5.4.1 Βασικοί ορισμοί

Αν και πολλές φορές ο όρος έχει επικρατήσει να χαρακτηρίζει παράλληλες μηχανές, εντούτοις μπορεί να χρησιμοποιηθεί και στην περίπτωση που έχουμε απλώς ένα δίκτυο από υπολογιστές που με την βοήθεια κατάλληλου λογισμικού συνεργάζονται μεταξύ τους. Σε αυτό το κείμενο δεν έχουμε σκοπό να ασχοληθούμε με το υλικό τέτοιων συστημάτων αλλά με το λογισμικό τους.

5.4.2 Sockets

Σε κάθε περίπτωση που θέλουμε να κάνουμε υπολογιστές να συνεργάζονται μεταξύ τους χρειαζόμαστε μία δίοδο επικοινωνίας. Αυτή η δίοδος είναι οι sockets. Αυτά είναι ειδικού τύπου αρχεία που χρησιμοποιούν τις συσκευές δικτυώσεως των υπολογιστών, για να επιτύχουν ανταλλαγή δεδομένων ανάμεσα στους υπολογιστές. Σε αυτά τα αρχεία μπορούμε να γράφουμε και να διαβάζουμε. Για την επίτευξη επικοινωνίας χρειαζόμαστε κάποια πόρτα του υπολογιστή που να είναι ελεύθερη για την ανταλλαγή μηνυμάτων. Ελεύθερη σημαίνει πως δεν απασχολείται από κάποια άλλη εργασία, όπως η πόρτα 23 για το πρόγραμμα telnet.

[1] shared memory

5.5 MPI

Η πλέον διαδεδομένη προγραμματιστική πλατφόρμα για υλοποίηση παράλληλων προγραμμάτων είναι η MPI[2] και με αυτήν θα ασχοληθούμε στο υπόλοιπο αυτού του κεφαλαίου. Η βασική φιλοσοφία της MPI είναι η ύπαρξη κάποιας ιεραρχίας στις συνεργαζόμενες διεργασίες και η ανταλλαγή μηνυμάτων με την χρήση ειδικών μηχανισμών που ονομάζονται communicators.

5.5.1 Διάλεκτοι

Λόγω της μεγάλης διαδόσεως που γνώρισε η MPI υπήρξαν αρκετές υλοποιήσεις της βιβλιοθήκης που οι περισσότερες είναι δωρεάν. Ωστόσο προγράμματα, τα οποία είναι γραμμένα σε μία διάλεκτο μπορούν να μεταφερθούν και στις άλλες διαλέκτους, αρκεί να αρχικοποιηθεί το σύστημα με τον τρόπο που απαιτεί η συγκεκριμένη διάλεκτος. Γενικά οι διάλεκτοι που υπάρχουν και είναι ελεύθερα διαθέσιμες είναι:

MPICH Δημιουργήθηκε από το Argonne National Lab του πανεπιστημίου της πολιτείας του Μισσισίπι. Ο δικτυακός της τόπος είναι info.mcs.anl.gov

CHIMP Δημιουργήθηκε από το πανεπιστήμιου του Εδιμβούργου και ο δικτυακός της τόπος είναι ftp.epcc.ed.ac.uk

LAM Είναι η πλέον διαδεδομένη διάλεκτος και είναι αυτή που θα χρησιμοποίησουμε στις εφαρμογές μας. Δημιουργήθηκε από το κέντρο υπερυπολογιστών του Οχάιο και ο δικτυακός της τόπος είναι tbag.osc.edu

5.5.2 Αρχικοποίηση συστήματος

Η αρχικοποίηση ενός Lam συστήματος απαιτεί την ύπαρξη ενός αρχείου με τα ονόματα των μηχανημάτων τα οποία θα συμμετάσχουν στην εκτέλεση του παράλληλου προγράμματος. Για να ξεκινήσει το σύστημα το σύστημα στο οποίο γίνεται η εκκίνηση θα πρέπει να υπάρχει στην λίστα των μηχανημάτων. Η σειρά των μηχανημάτων είναι σημαντική, καθώς σε κάθε μηχάνημα ανατίθεται μία ετικέτα κατά το πρότυπο n<αριθμός> όπου το n προέρχεται από το node. Ένα παράδειγμα τέτοιου αρχείου είναι το ακόλουθο:

```
195.130.121.245
clio.cs.uoi.gr
pontus.cs.uoi.gr
priapus.cs.uoi.gr
```

Μπορούμε να χρησιμοποιήσουμε IP αλλά και διευθύνσεις ονόματος σε αυτό το αρχείο. Εξαιτίας κάποιου bug στην υλοποίηση σε Linux θα πρέπει το όνομα του μηχανήματος εκκινήσεως να δίνεται σε IP. Αφού γίνει αυτό μπορούμε να ελέγξουμε αν μπορεί να δουλέψει η παραπάνω τοπολογία με το πρόγραμμα recon. Το βήμα αυτό δεν είναι απαραίτητο, αλλά μπορεί να μας προλάβει από λάθη, όπως

[2]Message Passing Interface

στην περίπτωση που κάποιο μηχάνημα της λίστας είναι εκτός δικτύου. Δίνοντας recon -v lamhosts θα πάρουμε την ακόλουθη έξοδο:

```
recon: testing n0 (195.130.121.245)
recon: testing n1 (clio.cs.uoi.gr)
recon: testing n2 (pontus.cs.uoi.gr)
recon: testing n3 (priapus.cs.uoi.gr)
```

Αν δεν εμφανιστεί κανένα λάθος ξεκινάμε την τοπολογία δίνοντας:

lamboot -v lamhosts

Σε κάθε μηχάνημα της λίστας θα τοποθετηθεί από ένας server που θα ακούει σε μία συγκεκριμένη πόρτα για τα μηνύματα που θα περνάνε ανάμεσα στην ομάδα των μηχανημάτων.

5.5.3 Μεταγλώττιση και εκτέλεση

Για να μεταγλωττίσουμε ένα πρόγραμμα σε LAM θα πρέπει να χρησιμοποιήσουμε ειδικές εκδόσεις μεταγλωττιστών που παρέχονται με την βιβλιοθήκη. Η βιβλιοθήκη δουλεύει σε Fortran77, C, C++ και επομένως μπορεί να καλύψει το σύνολο σχεδόν των προγραμματιστών. Επίσης η διαπροσωπεία της βιβλιοθήκης ελάχιστα διαφέρει από γλώσσα σε γλώσσα. Οι ειδικοί μεταγλωττιστές που παρέχονται είναι:

- hcc (Για την C).
- hcp (Για την C++)
- hf77 (Για την Fortran77)

Αφού μεταγλωττίσουμε το πρόγραμμά σε κάθε μηχανή που μετέχει στην τοπολογία δεν μένει παρά να το ξεκινήσουμε δίνοντας:

mpirun -np N program

Όπου N είναι ο αριθμός των διεργασιών που θέλουμε να εκτελεστούν. Αφού γίνει αυτό σε κάθε μηχάνημα με την σειρά ξεκινά η εκτέλεση του παράλληλου προγράμματος. Αν δεν θέλουμε να γίνει η εκτέλεση σε όλα τα μηχανήματα μπορούμε να προσδιορίσουμε σε ποιο μηχάνημα θα γίνει η εκτέλεση με τον εξής τρόπο:

mpirun n2-4 -np N program

Η παραπάνω γραμμή λέει πως το program θα εκτελεστεί μόνον στην τρίτη, την τέταρτη και την πέμπτη μηχανή της τοπολογίας και συνολικά θα υπάρξουν N διεργασίες. Φυσικά αν οι διεργασίες είναι περισσότερες από τις μηχανές (χωρίς αυτό να συνίσταται) τότε σε κάποιες μηχανές θα υπάρξουν περισσότερες από μία διεργασίες.

5.5.4 Τύποι δεδομένων

Στην MPI εμφανίζονται σχεδόν όλοι οι τύποι που συναντάμε στις κλασσικές γλώσσες προγραμματισμού, συμβολιζόμενες με σταθερές προκειμένου να πετύχουμε μεταφερσιμότητα των προγράμματών μας[10].

5.5.5 Μεταγωγείς μηνυμάτων

Για να υπάρξει ανταλλαγή μηνυμάτων μεταξύ διεργασιών πρέπει να χρησιμοποιηθεί κάποια δίοδος επικοινωνίας. Αυτή η δίοδος επικοινωνίας ονομάζεται communicator και για να δουλέψει χρειάζεται την διεπαφή δικτύου που υπάρχει στους συνδεόμενους υπολογιστές. Αυτή η διεπαφή δικτύου δεν έχει σε τίποτα να κάνει με την MPI και μπορεί να είναι κάποια DIALUP σύνδεση ή και κάποια T1 σύνδεση. Μέσω αυτής της διεπαφής η MPI δημιουργεί sockets για την επικοινωνία των υπολογιστών.

Όταν το σύστημα ξεκινά την εργασία του μία μόνο socket δημιουργείται για την διαβίβαση μηνυμάτων και αυτήν παρίσταται με την συμβολική σταθερά MPI_COMM_WORLD. Συνήθως αυτήν θα χρησιμοποιούμε στα προγράμματά μας, χωρίς αυτό να σημαίνει πως σε συγκεκριμένες περιπτώσεις δεν θα χρειαστούμε και άλλες διόδους επικοινωνίας. Αν για παράδειγμα έχουμε διαφορετικούς τύπους μηνυμάτων, κάθε τύπος μπορεί να χρησιμοποιεί διαφορετικό μεταγωγέα. Βέβαια κάθε ένας μεταγωγέας σημαίνει ένα ακόμα ανοικτό αρχείο και πάντα υπάρχει περιορισμός στο πλήθος των ανοικτών αρχείων.

Γενικά στα προγράμματα MPI υπάρχουν δύο τρόποι επικοινωνίας ανάμεσα στις διεργασίες:

1. **Επιλεκτική επικοινωνία**. Σε αυτήν την περίπτωση υπάρχει άμεση αποστολή μηνύματος από μία διεργασία προς κάποια άλλη.

2. **Broadcast επικοινωνία**. Σε αυτήν την περίπτωση μία διεργασία στέλνει ένα μήνυμα προς όλες τις υπόλοιπες διεργασίες.

Κεφάλαιο 6

Το Προτεινόμενο Μοντέλο

Σε αυτό το κεφάλαιο περιγράφεται το μοντέλο που χρησιμοποιήθηκε για την προσέγγιση των συναρτήσεων. Το κεφάλαιο ξεκινά με κάποιους μαθηματικούς ορισμούς και στην συνέχεια επεκτείνεται στην προγραμματιστική υλοποίηση του μοντέλου.

6.1 Ορισμοί

6.1.1 Θεώρηση

Μία βασική αρχή που μαθαίνει κάποιος ασχολούμενος με την πληροφορική, είναι η διαμέριση ενός δύσκολου προβλήματος σε πολλά μικρότερα και επίλυση του καθενός από αυτά τα μικρά προβλήματα. Αυτή η βασική αρχή έχει επικρατήσει να ονομάζεται **top-down** τεχνική επειδή μοιάζει να αντιμετωπίζουμε το πρόβλημα από την κορυφή προς την βάση του. Αυτή η τεχνική μπορεί άνετα να εφαρμοστεί και στην προσέγγιση συναρτήσεων, κάτι που το είδαμε και στο πρώτο κεφάλαιο αυτού του κειμένου. Εκεί είδαμε ειδικά πολυώνυμα που άλλοτε ήταν πολυώνυμα Lagrange και άλλοτε splines. Ανάλογα με την δυσκολία ενός δεδομένου προβλήματος, αυτά τα εργαλεία έχουν αποδειχθεί εξαιρετικά χρήσιμα. Ωστόσο υπάρχουν και πολύ δύσκολα προβλήματα στα οποία η χρήση αυτών των πολυωνύμων δεν δίνει ικανοποιητικά αποτελέσματα. Επίσης η παράλληλη λύση με την χρήση των παραπάνω τμηματικών πολυωνύμων δεν είναι εφικτή και έτσι δεν μπορούμε να επωφεληθούμε από τα πλεονεκτήματα του παράλληλου υπολογισμού που είδαμε στο προηγούμενο κεφάλαιο.

6.1.2 Γενική περιγραφή της μεθόδου

Για να μπορέσουμε να κατανοήσουμε καλύτερα την μέθοδο που θα ακολουθήσουμε, ας δούμε σε πιο αυστηρή μορφή το θεμελιώδες πρόβλημα της προσεγγίσεως συναρτήσεων.

> Δοθέντων M σημείων με τιμές (x_i, y_i) i=1..M, με τα σημεία $x_i \in R^N$ θέλουμε να σχεδιάσουμε μία λεία υπερεπιφάνεια που να είναι όσο πιο κοντά μπορούμε σε αυτά τα σημεία.

Ο συνηθισμένος τρόπος να επιλυθεί το παραπάνω είναι με την θεώρηση ενός παραμετρικού μοντέλου Ψ(x,p). Σε αυτό το μοντέλο διαμορφώνουμε κατάλληλα τις παραμέτρους p, έτσι ώστε να ελαχιστοποιήσουμε το συνολικό τετραγωνικό σφάλμα που δίδεται από τον τύπο:

$$E[p] = \sum_{i=1}^{M} [\Psi(x_i, p) - y_i]^2 \tag{6.1}$$

Αν όμως διαμερίσουμε τον χώρο D του προβλήματός μας σε μικρότερους χώρους D_i τότε σε κάθε χώρο μπορούμε να θέσουμε ένα μοντέλο με διαφορετικές παραμέτρους, το οποίο υποθέτουμε πως θα έχει καλύτερες προσεγγιστικές ικανότητες από το μεγαλύτερο μιας και θα εφαρμόζεται σε μικρότερο πλήθος σημείων. Για να μπορέσουμε να το εξασφαλίσουμε αυτό θα πρέπει να διαμερίσουμε τον χώρο του προβλήματος, έτσι ώστε να μην υπάρχει υπερκάλυψη μεταξύ των διαφορετικών υποχώρων και επιπλέον να τηρούνται και κάποιες συνθήκες συνέχειας στα όρια αυτών των περιοχών. Για παράδειγμα δεν θα θέλαμε στο ίδιο σημείο δύο διαφορετικά μοντέλα να δίνουν διαφορετικές τιμές. Όπως θα δούμε πολύ σύντομα οι οριακές συνθήκες μπορεί να είναι ακόμα πιο περίπλοκες. Το μοντέλο σε κάθε χώρο D_i έχει την ετικέτα $\psi_i(x, p^i, q^i)$ με το διάνυσμα p^i να είναι το διάνυσμα παραμέτρων για το συγκεκριμένο μοντέλο και το διάνυσμα q^i να είναι το διάνυσμα των οριακών συνθηκών για το μοντέλο αυτό με το οποίο εξασφαλίζουμε συνέχεια τιμής ή/και παραγώγων ανάμεσα σε γειτονικά μοντέλα. Πρέπει να επισημανθεί πως το διάνυσμα p^i εξαρτάται μόνο από το δεδομένο μοντέλο, ενώ το q^i θα εξαρτάται και από γειτονικά μοντέλα. Με δεδομένο το παραπάνω μοντέλο το τετραγωνικό σφάλμα σε κάθε υποδιάστημα μπορεί να εκφραστεί από τον τύπο:

$$E_i\left[p^i, q^i\right] = \sum_{x_k \in D_i} \left[\psi_i\left(x_k, p^i, q^i\right) - y_k\right]^2 \tag{6.2}$$

Το συνολικό σφάλμα επομένως μπορεί να δοθεί από τον τύπο

$$E[p, q] = \sum_i E_i[p^i, q^i] \tag{6.3}$$

Το διάνυσμα παραμέτρων q^i είναι τέτοιο ώστε να ισχύει πάντα:

$$\Psi(x, p) = \psi_i(x, p^i, q^i),\ x \in D_i \tag{6.4}$$

Πριν φτάσουμε να ορίσουμε το μοντέλο μας και την διαδικασία εκπαιδεύσεως αυτού, καλό θα ήταν να δούμε δύο πολύ σημαντικές έννοιες, όπως είναι τα πολυώνυμα Obreshkov και τα Neural Splines.

6.1.3 Πολυώνυμα Obreshkov

Ας θεωρήσουμε μία συνεχώς παραγωγίσιμη συνάρτηση $f(x)$ με το $x \in [a,b]$ και ένα πολυώνυμο $P^{k,m}_{a,b}(f,x)$ με τις ακόλουθες ιδιότητες:

$$\begin{array}{l} \frac{\partial^j}{\partial x^j} P^{k,m}_{a,b}(f,a) = \frac{\partial^j}{\partial x^j} f(x)\mid_{x=a} \equiv f^{(j)}_a,\ \forall j = 0,1,...,k \\ \frac{\partial^j}{\partial x^j} P^{k,m}_{a,b}(f,b) = \frac{\partial^j}{\partial x^j} f(x)\mid_{x=b} \equiv f^{(j)}_b,\ \forall j = 0,1,...,m \end{array} \tag{6.5}$$

Δηλαδή παραγωγίζοντας αυτά τα πολυώνυμα στα άκρα του διαστήματος λαμβάνουμε τις αντίστοιχες παραγώγους της συναρτήσεως σε αυτά τα σημεία. Φυσικά αλλάζοντας τους όρους k, m καταλαβαίνουμε πως μπορούμε να έχουμε όποιας τάξεως παραγώγων συνέχεια θέλουμε. Βέβαια για να συμβαίνει αυτό το πολυώνυμο θα πρέπει να έχει βαθμό τουλάχιστον m+k+1. Ο Obrehkov έδωσε τον ακόλουθο τύπο για τα πολυώνυμα με την ιδιότητα 6.5 και με τον ελάχιστο βαθμό m+k+1[12]:

$$P^{k,m}_{a,b}(f,x) = \sum_{j=0}^{k} f^{(j)}(a)\frac{(x-b)^{(m+1)}(x-a)^j}{j!(a-b)^{m+1}} \sum_{i=0}^{k-j} \left(\begin{array}{c} m+i \\ i \end{array} \right) \frac{(x-a)^i}{(b-a)^i} + \sum_{j=0}^{m} f^{(j)}(b)\frac{(x-a)^{(k+1)}(x-b)^j}{j!(b-a)^{(k+1)}} \sum_{i=0}^{m-j} \left(\begin{array}{c} k+i \\ i \end{array} \right) \frac{(x-b)^i}{(a-b)^i}$$

Αρκεί να παραγωγίσουμε το παραπάνω πολυώνυμο τον απαιτούμενο αριθμό φορών, ώστε να διαπιστώσουμε πως ισχύει η εξίσωση 6.5.

6.1.4 Neural Splines

Αφού ορίσαμε τα πολυώνυμα είναι ώρα να ορίσουμε και τους τελεστές. Ορίζουμε τον τελεστή $L^{k,m}_{x\in[a,b]}$ ως εξής:

$$L^{k,m}_{x\in[a,b]} f(x) = P^{k,m}_{a,b}(f,x) \tag{6.6}$$

Και έτσι μπορούμε να ορίσουμε την ποσότητα $S(f,x)$ ως:

$$S(f,x) = (1 - L^{k,m}_{x\in[a,b]})f(x) = f(x) - P^{k,m}_{a,b}(f,x) \tag{6.7}$$

και την ποσότητα $B(f,x)$ ως:

$$B(f,x) = (1 - (1 - L^{k,m}_{x\in[a,b]}))f(x) = f(x) - S(f,x) = P^{k,m}_{a,b}(f,x) \tag{6.8}$$

Η συνάρτηση $S(f,x)$ έχει εξ ορισμού την ιδιότητα πως μηδενίζεται στα άκρα του διαστήματος, όπως και όλες οι παράγωγοι της. Επειδή βασίζεται στην συνάρτηση f μπορούμε να την ονομάσουμε f-spline. Από την άλλη η συνάρτηση $B(f,x)$ στα άκρα του διαστήματος ισούται με την συνάρτηση f όχι μόνο ώς προς την τιμή,αλλά και ως προς τις παραγώγους.

Στην περίπτωση προβλημάτων στις δύο διαστάσεις κάθε υποδιάστημα μπορεί να γραφεί σαν $[a_1, b_1] \times [a_2, b_2]$ και οι συναρτήσεις $S(f, x)$ και $B(f, x)$ μπορούν να γραφούν ως ακολούθως:

$$S(f, x_1, x_2) = (1 - L^{k_1,m_1}_{x_1 \in [a_1,b_1]})(1 - L^{k_2,m_2}_{x_2 \in [a_2,b_2]}) f(x_1, x_2) \tag{6.9}$$

και

$$B(f, x_1, x_2) = f(x_1, x_2) - S(f, x_1, x_2) =$$

$$= (L^{k_1,m_1}_{x_1 \in [a_1,b_1]} + L^{k_2,m_2}_{x_2 \in [a_2,b_2]} - L^{k_1,m_1}_{x_1 \in [a_1,b_1]} L^{k_2,m_2}_{x_2 \in [a_2,b_2]}) f(x_1, x_2) \tag{6.10}$$

Αν διαλέξουμε την συνάρτηση f(x) να είναι ένα νευρωνικό, έστω το N(x), τότε η συνάρτηση S(f,x) αποκαλείται Neural Spline. Σε κάθε υποχώρο D_i το μοντέλο που έχουμε διατυπώνεται με την εξίσωση:

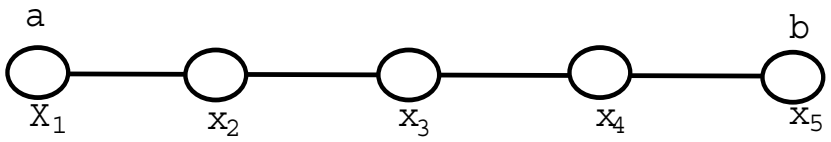
$$\psi_i(x, p^i, q^i) = B(f, x) + S(N, x) \tag{6.11}$$

Όπου $N(x, p^i)$ είναι το νευρωνικό στον υποχώρο D_i με το διάνυσμα παραμέτρων p^i. Το διάνυσμα q^i περιέχει σίγουρα τις προσεγγίσεις των τιμών της $f(x)$ στα όρια του διαστήματος και πιθανόν τις τιμές των προσεγγίσεις παραγώγων κάποιας τάξεως σε αυτά τα σημεία.

6.2 Μοντέλα μίας διαστάσεως

6.2.1 Διαμέριση ευθείας

Στην μονοδιάστατη περίπτωση το πεδίο ορισμού της συναρτήσεως που θέλουμε να προσεγγίσουμε είναι μία ευθεία. Άρα αν θέλουμε να διαχωρίσουμε ένα διάστημα σε τμήματα και να βάλουμε ένα ξεχωριστό μοντέλο σε κάθε διάστημα χρειαζόμαστε να διαχωρίσουμε την ευθεία σε ευθύγραμμα τμήματα.



Σχήμα 6.1: Ομοιόμορφη διαμέριση ευθείας σε τέσσερα διαστήματα

Στο σχήμα χωρίζουμε το διάστημα $[a, b]$ σε τέσσερα υποδιαστήματα: $[x_0, x_1]$, $[x_1, x_2]$, $[x_2, x_3]$, και $[x_3, x_4]$. Σε κάθε υποδιάστημα τοποθετούμε και ένα διαφορετικό μοντέλο με διαφορετικές παραμέτρους. Ωστόσο καθώς βλέπουμε τα διαστήματα ενώνονται στα άκρα τους και επομένως πρέπει να καθορίσουμε κάποιες οριακές συνθήκες ανάμεσα στα διαφορετικά μοντέλα. Όπως θα δούμε στην συνέχεια αυτές οι οριακές συνθήκες μπορεί να είναι απλώς συνέχεια τιμής ή/και συνέχεια παραγώγων.

6.2.2 Απλό μοντέλο

Το απλούστερο μοντέλο που μπορούμε να κατασκευάσουμε στην μία διάσταση είναι όταν δεν απαιτείται συνέχεια παραγώγων, παρά μόνον τιμής. Σε αυτήν την περίπτωση (και μετά από πράξεις στην Εξίσωση (6.11)) καταλήγουμε στο επόμενο μοντέλο:

$$\begin{aligned}\psi_i(x,p,q) = f^{(0)}(a)\tfrac{x-b}{a-b} + f^{(0)}(b)\tfrac{x-a}{b-a} + N(x,p) - \\ \left[N(a,p)\tfrac{x-b}{a-b} + N(b,p)\tfrac{x-a}{b-a}\right]\end{aligned} \tag{6.12}$$

Αν κάνουμε κάποιους υπολογισμούς θα βρούμε πως $\psi_i(a,p,q) = f^{(0)}(a)$ και $\psi_i(b,p,q) = f^{(0)}(b)$, κάτι που είναι αναμενόμενο μετά την θεωρία που διατυπώσαμε στις προηγούμενες παραγράφους. Οι παράμετροι $f^{(0)}(a)$ και $f^{(0)}(b)$ δεν είναι οι πραγματικές τιμές της συναρτήσεως f στα σημεία a και b αλλά προσεγγίσεις αυτών, καθώς εκ των πραγμάτων δεν μπορούν να είναι πάντα γνωστά κατά την κατασκευή του μοντέλου. Αυτές οι παράμετροι ονομάζονται εξωτερικές παράμετροι σε αντιδιαστολή με τις παραμέτρους των νευρωνικών δικτύων που έχει κάθε μοντέλο. Για λόγους ευκολίας το παραπάνω μοντέλο θα το ονομάζουμε ΜΟΝΤΕΛΟ - 0.

6.2.3 Μοντέλο με συνέχεια παραγώγων

Αν ανάμεσα στα διαφορετικά μοντέλα απαιτήσουμε συνέχεια παραγώγων τότε το μοντέλο που προκύπτει έχει ως εξής:

$$\begin{aligned}\psi(x,p,q) = \quad & f^{(0)}(a)\pi_{3,0}(x,a,b) + f^{(1)}(a)\pi_{3,1}(x,a,b) \\ & + f^{(0)}(b)\tau_{3,0}(x,a,b) + f^{(1)}(b)\tau_{3,1}(x,a,b) \\ & + N(x,p) - [N(a,p)\pi_{3,0}(x,a,b) + N^{(1)}(a,p)\pi_{3,1}(x,a,b)) \\ & + N(b,p)\tau_{3,0}(x,a,b) + N^{(1)}(b,p)\tau_{3,1}(x,a,b)]\end{aligned} \tag{6.13}$$

Με τα πολυώνυμα $\pi_{3,0}$,$\pi_{3,1}$,$\tau_{3,0}$,$\tau_{3,1}$ να ορίζονται ως ακολούθως :

$$\pi_{3,0}(x,a,b) = \frac{(x-b)^2}{(a-b)^2}(1 + 2\frac{x-a}{b-a}) \tag{6.14}$$

$$\pi_{3,1}(x,a,b) = (x-a)\frac{(x-b)^2}{(a-b)^2} \tag{6.15}$$

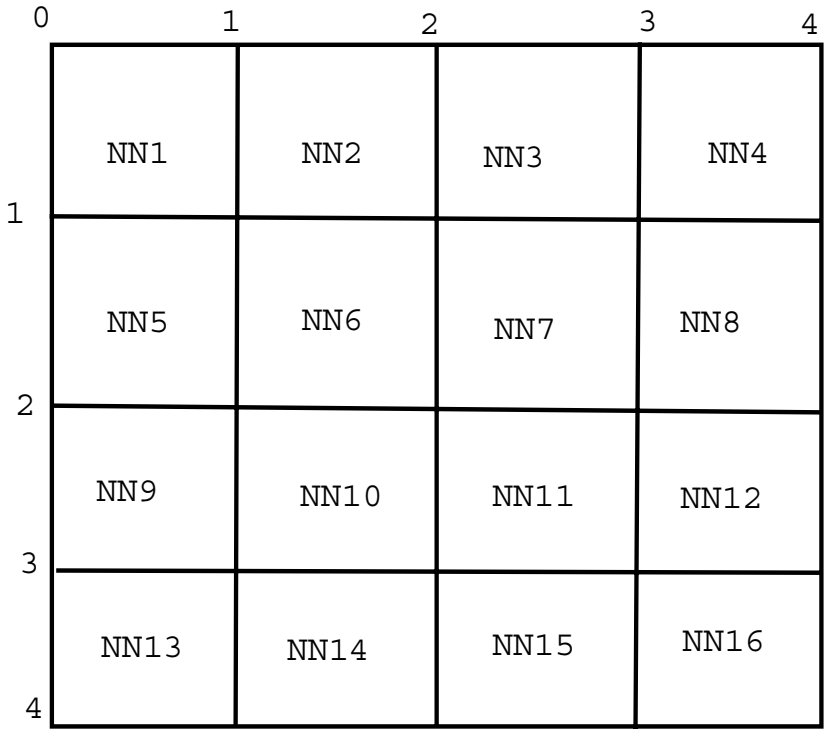
$$\tau_{3,j}(x,a,b) = \pi_{3,j}(x,b,a) \tag{6.16}$$

Και σε αυτό το μοντέλο, όπως και στο προηγούμενο οι παράμετροι $f^{(0)}(a)$,$f^{(0)}(b)$ είναι προσεγγίσεις της τιμής της συναρτήσεως στα δεδομένα σημεία. Οι παράμετροι $f^{(1)}(a)$,$f^{(1)}(b)$ είναι προσεγγίσεις της τιμής της παραγώγου στα δεδομένα σημεία. Έτσι οι εξωτερικές παράμετροι του δεύτερου μοντέλου είναι διπλάσιες σε ποσότητα από αυτές του πρώτου μοντέλου. Για λόγους ευκολίας το μοντέλο της εξισώσεως (6.13) θα το ονομάζουμε ΜΟΝΤΕΛΟ - 1.

6.3 Μοντέλο δύο διαστάσεων

6.3.1 Διαμέριση πλέγματος

Όταν τα σημεία εισόδου ορίζονται στην μία διάσταση, τότε διαχωρίζουμε ένα ευθύγραμμο τμήμα σε επιμέρους ευθύγραμμα τμήματα. Όταν έχουμε δύο διαστάσεις έχουμε να διαχωρίσουμε έναν ορθογώνιο χώρο σε επιμέρους ορθογώνιο. Οπτικά το αποτέλεσμα της διαμερίσεως δημιουργεί ένα πλέγμα. Αυτή η κατάσταση παρουσιάζεται στο επόμενο σχήμα :



Σχήμα 6.2: Ομοιόμορφη διαμέριση ορθογωνίου στις δύο διαστάσεις σε 16 τμήματα

Στο Σχήμα 6.2 χωρίζουμε τον χώρο σε 16 τμήματα χρησιμοποιώντας μία διαμέριση 5×5. Δεν είναι απαραίτητο για το μοντέλο προσεγγίσεως να χρησιμοποιούμε ομοιόμορφη διαμέριση, όπως δεν είναι απαραίτητο να χρησιμοποιούμε ισαπέχουσες διαμερίσεις. Ωστόσο στην παρούσα μορφή του το πρόγραμμα που υλοποιεί το μοντέλο χρησιμοποιεί μόνον ισαπέχουσες διαμερίσεις.

6.3.2 Περιγραφή μοντέλου

Στις δύο διαστάσεις δεν χρησιμοποιούμε συνέχεια παραγώγων ανάμεσα στα μοντέλα κάθε διαστήματος παρά μόνον συνέχεια τιμής. Αν θέσουμε $k_1 = k_2 =$

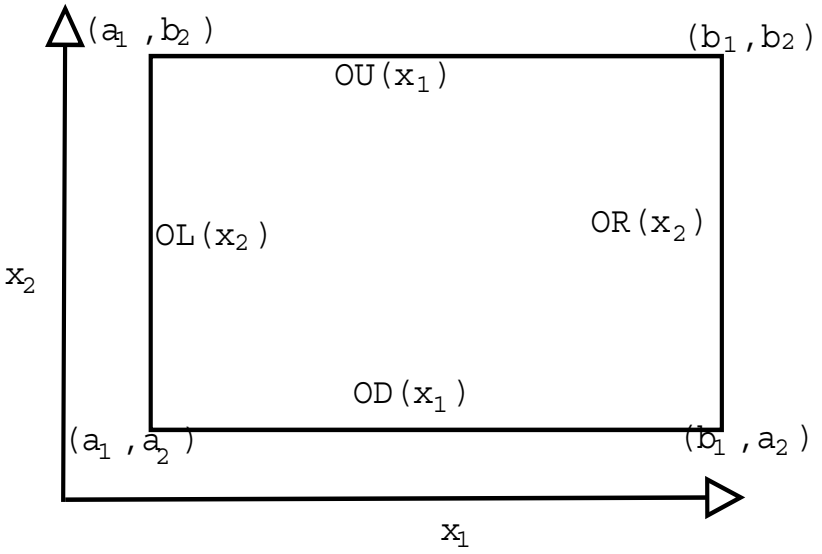
$m_1 = m_2 = 0$ στις εξισώσεις (6.9) και (6.10) τότε θα πάρουμε διαδοχικά τις επόμενες εξισώσεις :

$$\begin{aligned} B(f,x_1,x_2) = \quad & f(a_1,x_2)\tfrac{x_1-b_1}{a_1-b_1} + f(b_1,x_2)\tfrac{x_1-a_1}{b_1-a_1} \\ & + f(x_1,a_2)\tfrac{x_2-b_2}{a_2-b_2} + f(x_1,b_2)\tfrac{x_2-a_2}{b_2-a_2} \\ & - \{[f(a_1,a_2)\tfrac{x_1-b_1}{a_1-b_1} + f(b_1,a_2)\tfrac{x_1-a_1}{b_1-a_1}]\tfrac{x_2-b_2}{a_2-b_2} \\ & + [f(a_1,a_2)\tfrac{x_1-b_1}{a_1-b_1} + f(x_1,b_2)\tfrac{x_1-a_1}{b_1-a_1}]\tfrac{x_2-a_2}{b_2-a_2}\} \end{aligned} \tag{6.17}$$

Και

$$\begin{aligned} S(N,x_1,x_2) = \quad & N(x_1,x_2) - N(x_1,a_2)\tfrac{x_2-b_2}{a_2-b_2} - N(x_1,b_2)\tfrac{x_2-a_2}{b_2-a_2} \\ & - N(a_1,x_2)\tfrac{x_1-b_1}{a_1-b_1} - N(b_1,x_2)\tfrac{x_1-a_1}{b_1-a_1} \\ & + \tfrac{x_2-b_2}{a_2-b_2}\left(N(a_1,a_2)\tfrac{x_1-b_1}{a_1-b_1} + N(b_1,a_2)\tfrac{x_1-a_1}{b_1-a_1}\right) \\ & + \tfrac{x_2-a_2}{b_2-a_2}\left(N(a_1,b_2)\tfrac{x_1-b_1}{a_1-b_1} + N(b_1,b_2)\tfrac{x_1-a_1}{b_1-a_1}\right) \end{aligned} \tag{6.18}$$

Στην συνάρτηση $B(f,x_1,x_2)$ η συνάρτηση f δεν είναι η συνάρτηση που θέλουμε να προσεγγίσουμε, αλλά ένα μοντέλο αυτής το οποίο είναι ξεχωριστό σε κάθε άκρο του ορθογωνίου στο οποίο έχουμε το συγκεκριμένο μοντέλο, σύμφωνα με το επόμενο σχήμα:



Σχήμα 6.3: Πολυώνυμα Obreshkov δύο διαστάσεων

Τα πολυώνυμα $OL(x_2)$,$OR(x_2)$ μεταβάλλονται μόνον ως προς την διάσταση x_2 και τα πολυώνυμα $OU(x_1)$,$OD(x_1)$ ως προς την διάσταση x_1. Τα τέσσερα

παραπάνω πολυώνυμα ορίζονται σύμφωνα με τους ακόλουθους τύπους:

$$\begin{aligned} OL(x_2) \quad = \quad & f^{(0)}(b_{1,}a_2)\pi_{5,0}(x_2,a_2,b_2) + f^{(0)}(b_1,b_2)\tau_{5,0}(x_2,a_2,b_2) + \\ & f^{(1)}_{x_2}(b_1,a_2)\pi_{5,1}(x_2,a_2,b_2) + f^{(1)}_{x_2}(b_1,b_2)\tau_{5,1}(x_2,a_2,b_2) + \\ & f^{(2)}_{x_2}(b_1,a_2)\pi_{5,2}(x_2,a_2,b_2) + f^{(2)}_{x_2}(b_1)\tau_{5,2}(x_2,a_2,b_2) \end{aligned} \tag{6.19}$$

$$\begin{aligned} OR(x_2) \quad = \quad & f^{(0)}(a_1,a_2)\pi_{5,0}(x_2,a_2,b_2) + f^{(0)}(a_1,b_2)\tau_{5,0}(x_2,a_2,b_2) + \\ & f^{(1)}_{x_2}(a_1,a_2)\pi_{5,1}(x_2,a_2,b_2) + f^{(1)}_{x_2}(a_1,b_2)\tau_{5,1}(x_2,a_2,b_2) + \\ & f^{(2)}_{x_2}(a_1,a_2)\pi_{5,2}(x_2,a_2,b_2) + f^{(2)}_{x_2}(a_1,b_2)\tau_{5,2}(x_2,a_2,b_2) \end{aligned} \tag{6.20}$$

$$\begin{aligned} OU(x_1) \quad = \quad & f^{(0)}(a_1,a_2)\pi_{5,0}(x_1,a_1,b_1) + f^{(0)}(b_1,a_2)\tau_{5,0}(x_1,a_1,b_1) + \\ & f^{(1)}_{x_1}(a_1,a_2)\pi_{5,1}(x_1,a_1,b_1) + f^{(1)}_{x_1}(b_1,a_2)\tau_{5,1}(x_1,a_1,b_1) + \\ & f^{(2)}_{x_1}(a_1,a_2)\pi_{5,2}(x_1,a_1,b_1) + f^{(2)}_{x_1}(b_1,a_2)\tau_{5,2}(x_1,a_1,b_1) \end{aligned} \tag{6.21}$$

$$\begin{aligned} OD(x_1) \quad = \quad & f^{(0)}(a_1,b_2)\pi_{5,0}(x_1,a_1,b_1) + f^{(0)}(a_2,b_2)\tau_{5,0}(x_1,a_1,b_1) + \\ & f^{(1)}_{x_1}(a_1,b_2)\pi_{5,1}(x_1,a_1,b_1) + f^{(1)}_{x_1}(a_2,b_2)\tau_{5,1}(x_1,a_1,b_1) + \\ & f^{(2)}_{x_1}(a_1,b_2)\pi_{5,2}(x_1,a_1,b_1) + f^{(2)}_{x_1}(a_2,b_2)\tau_{5,2}(x_1,a_1,b_1) \end{aligned} \tag{6.22}$$

Τα παραπάνω πολυώνυμα πέμπτου βαθμού χρησιμεύουν για τον υπολογισμό των συναρτήσεων $f(a_1,x_2)$,$f(b_1,x_2)$,$f(x_1,a_2)$,$f(x_1,b_2)$ που δεν θα μπορούσαν να θεωρηθούν παράμετροι του συστήματος, καθώς μεταβάλλονται. Τα πολυώνυμα π και τ μπορούν εύκολα να υπολογιστούν από τους τύπους υπολογισμού των πολυωνύμων όπως κάναμε και στην μία διάσταση. Πρέπει να σημειωθεί πως αν και έχουμε πολυώνυμα Obreshkov πέμπτου βαθμού απαιτούμε συνέχεια μόνον ως προς την τιμή. Οι πρόσθετοι παράμετροι στις παραπάνω εξισώσεις αποτελούν προσεγγίσεις μερικών παραγώγων στα αντίστοιχα σημεία σύμφωνα με τον ακόλουθο πίνακα, αλλά δεν απαιτούμε συνέχεια για αυτές τις παραμέτρους στα σημεία που ενώνονται τα ορθογώνια.

ΠΑΡΑΜΕΤΡΟΣ	ΠΡΟΣΕΓΓΙΖΕΙ
$f_{x_1}^{(1)}(a_1, a_2)$	$\frac{\partial f}{\partial x_1}(a_1, a_2)$
$f_{x_1}^{(1)}(a_1, b_2)$	$\frac{\partial f}{\partial x_1}(a_1, b_2)$
$f_{x_1}^{(1)}(b_1, a_2)$	$\frac{\partial f}{\partial x_1}(b_1, a_2)$
$f_{x_1}^{(1)}(b_1, b_2)$	$\frac{\partial f}{\partial x_1}(b_1, b_2)$
$f_{x_1}^{(2)}(a_1, a_2)$	$\frac{\partial^2 f}{\partial x_1^2}(a_1, a_2)$
$f_{x_1}^{(2)}(a_1, b_2)$	$\frac{\partial^2 f}{\partial x_1^2}(a_1, b_2)$
$f_{x_1}^{(2)}(b_1, a_2)$	$\frac{\partial^2 f}{\partial x_1^2}(b_1, a_2)$
$f_{x_1}^{(2)}(b_1, b_2)$	$\frac{\partial^2 f}{\partial x_1^2}(b_1, b_2)$
$f_{x_2}^{(1)}(a_1, a_2)$	$\frac{\partial f}{\partial x_2}(a_1, a_2)$
$f_{x_2}^{(1)}(a_1, b_2)$	$\frac{\partial f}{\partial x_2}(a_1, b_2)$
$f_{x_2}^{(1)}(b_1, a_2)$	$\frac{\partial f}{\partial x_2}(b_1, a_2)$
$f_{x_2}^{(1)}(b_1, b_2)$	$\frac{\partial f}{\partial x_2}(b_1, b_2)$
$f_{x_2}^{(2)}(a_1, a_2)$	$\frac{\partial^2 f}{\partial x_2^2}(a_1, a_2)$
$f_{x_2}^{(2)}(a_1, b_2)$	$\frac{\partial f^2}{\partial x_2^2}(a_1, b_2)$
$f_{x_2}^{(2)}(b_1, a_2)$	$\frac{\partial f^2}{\partial x_2^2}(b_1, a_2)$
$f_{x_2}^{(2)}(b_1, b_2)$	$\frac{\partial f^2}{\partial x_2^2}(b_1, b_2)$

Συνεπώς στις δύο διαστάσεις έχουμε περισσότερες εξωτερικές παραμέτρους από ότι στην μία διάσταση.

6.4 Κωδικοποίηση

Στην ενότητα αυτή θα εξετάσουμε τον τρόπο που κωδικοποιήθηκε το μοντέλο προσεγγίσεως που εξετάζουμε σε αυτό το κεφάλαιο. Πολλές φορές θα καταφύγουμε σε υπερβολικά μεγάλο βάθος λεπτομέρειας κώδικα, αλλά κάτι τέτοιο είναι απαραίτητο για την κατανόηση του συνόλου.

6.4.1 Αρχικοποίηση συστήματος

Το σύστημα για να ξεκινήσει χρειάζεται ένα βασικό αρχείο με το όνομα INIT. Αυτό λογικά θα πρέπει να βρίσκεται στον κατάλογο εκκινήσεως της εφαρμογής. Το αρχείο αποτελείται από αναθέσεις σε μεταβλητές. Κάθε ανάθεση βρίσκεται σε διαφορετική γραμμή. Μία τυπική μορφή που έχει το παραπάνω αρχείο είναι η ακόλουθη:

```
dimension= 1
dim=  2
nodes=  12
ipoints= 120
test_points= 1000
method=  0
eps=  0.00001
```

Με τις μεταβλητές του αρχείου να έχουν την ακόλουθη σημασία:

1. *dimension*. Καθορίζει την διάσταση των σημείων του προβλήματος.

2. *dim*. Θα υπάρχουν τόσες διαδοχικές εμφανίσεις αυτής της μεταβλητής,όσο είναι και η διάσταση των δεδομένων του προβλήματος. Κάθε εμφάνιση καθορίζει και ποιος είναι ο επιθυμητός διαχωρισμός σε αυτήν την διάσταση. Για παράδειγμα dim=2 και dim=3 καθορίζει πως θέλουμε δύο σημεία διαχωρισμού στην πρώτη διάσταση και 3 στην δεύτερη διάσταση. Με κατάλληλο υπολογισμό από αυτές τις μεταβλητές μπορούμε να βρούμε τα διαστήματα που θα έχουμε και επομένως και το πλήθος των νευρωνικών που θα χρησιμοποιηθούν. Ο τύπος είναι ο

$$intervals = \prod_{i=1}^{dimension} (dim_i - 1) \tag{6.23}$$

   Από τον παραπάνω τύπο και μόνον καταλαβαίνουμε πως $dim_i \geq 2$.

3. *nodes*. Δίνει τον αριθμό των κρυμμένων μονάδων που έχουμε σε κάθε νευρωνικό. Επομένως όλα τα νευρωνικά είναι ισοδύναμα, ως προς το πλήθος των κόμβων τους. Αν θέλουμε να υπολογίσουμε τον αριθμό των παραμέτρων που έχει το κάθε νευρωνικό μπορούμε να χρησιμοποιήσουμε τον τύπο:

$$N = (dimension + 2)nodes \tag{6.24}$$

4. *ipoints*. Είναι το συνολικό πλήθος των σημείων εκπαιδεύσεως. Δεν είναι το πλήθος των σημείων που βρίσκονται σε κάθε διάστημα. Σε κάθε διάστημα μπορεί να βρίσκονται από 0 έως και ipoints σημεία, καθώς ο διαχωρισμός που κάνουμε δεν εγγυάται πως θα υπάρξει ίδιο πλήθος σημείων σε κάθε διάστημα.

5. *test_points*. Είναι το συνολικό πλήθος των σημείων ελέγχου. Φυσιολογικά θα πρέπει να είναι περισσότερα από τα σημεία εκπαιδεύσεως, αν και κάτι τέτοιο δεν είναι υποχρεωτικό.

6. *method*. Είναι ένας αριθμός που καθορίζει και την βαθμό που θέλουμε να έχουν τα πολυώνυμα Obreshkov. Ο βαθμός των πολυωνύμων δίνεται από τον τύπο:

$$degree = 2method + 1 \tag{6.25}$$

7. *eps*. Είναι ένας μικρός αριθμός που καθορίζει την επιθυμητή ακρίβεια που θέλουμε από το σύστημά μας. Χρησιμοποιείται για να αλλάξει την μεταβλητή target του προγράμματος βελτιστοποιήσεως merlin.

Αφού αναλυθεί λεκτικά το παραπάνω αρχείο το σύστημα ψάχνει να βρει δύο αρχεία με ονόματα train.data και test.data. Το πρώτο αρχείο περιλαμβάνει τα πρότυπα εκπαιδεύσεως και το δεύτερο τα πρότυπα ελέγχου. Τα πρότυπα δεν υποτίθεται πως πρέπει να έχουν κάποια διάταξη σε αυτά τα αρχεία, εκτός του ότι το σημείο με χαμηλότερο X θα πρέπει να είναι το πρώτο σημείο στα αρχεία και το σημείο με το μεγαλύτερο X στο τέλος του αρχείου. Αν κάτι τέτοιο δεν συμβαίνει ένα απλό script σε Perl ή Awk μπορεί να λύσει το πρόβλημα. Το αρχείο train.data περιέχει τα ζευγάρια (σημείο, τιμή-συναρτήσεως). Από την άλλη το αρχείο test.data περιέχει την τριάδα (σημείο, τιμή-συναρτήσεως, τιμή-πρώτης-παραγώγου).

Πρέπει να επισημανθεί πως τα αρχεία INIT, train.data και test.data είναι τα ίδια σε κάθε μηχανή που μετέχει στην παράλληλη εκτέλεση του προγράμματος βελτιστοποιήσεως. Αυτό έγινε για να μειωθεί η πολυπλοκότητα του συστήματος και να μπορούμε να αλλάζουμε κατά την διάρκεια της εκτελέσεως το διάστημα εκπαιδεύσεως μίας διεργασίας.

6.4.2 Διαμέριση χώρου

Αφού το σύστημα εξετάσει τα αρχεία εκκινήσεως και καθορίσει κάποιες εσωτερικές μεταβλητές, θα πρέπει να δημιουργήσει τα διαστήματα μέσα στα οποία βρίσκονται τα πρότυπα κάθε νευρωνικού δικτύου και επομένως τα απαιτούμενα νευρωνικά δίκτυα. Ο αριθμός των διαστημάτων όπως είδαμε δίνεται από τον τύπο 6.23. Η σωστή διαμέριση του συστήματος προϋποθέτει πως το σημείο με το χαμηλότερο X θα βρίσκεται στην αρχή των σημείων και πως το σημείο με το μεγαλύτερο X στο τέλος των σημείων. Για να δημιουργηθούν τα διαστήματα χρειαζόμαστε επίσης δύο βασικές πληροφορίες:

1. **Offset**. Είναι ένας πίνακας που περιγράφει το «σημείο» στο οποίο βρίσκεται η μεγαλύτερη γωνία του διαστήματος. Αν για παράδειγμα είμαστε σε μία διάσταση και θέλουμε το offset για το δεύτερο διάστημα θα πάρουμε την τιμή 2. Ο τρόπος υπολογισμού αυτού του πίνακα περιγράφεται στην συνέχεια.

2. **Division**. Ο όρος αυτός είναι ένας πίνακας που περιγράφει την διαμέριση σε κάθε διάσταση του χώρου του προβλήματος. Ουσιαστικά συμπίπτει με τον πίνακα dim_i που είδαμε προηγουμένως.

6.4.3 Προεπεξεργασία

Αν τα μοντέλα προσεγγίσεως σε κάθε διάστημα ήταν διαφορετικά μεταξύ τους δεν θα υπήρχε καμία ανάγκη για το πρώτο βήμα εκτελέσεως που το ονομάσαμε προεπεξεργασία. Τα νευρωνικά δίκτυα θα εκπαιδεύοταν και δεν θα υπήρχε κανένα πρόβλημα. Από όσα είπαμε στο κεφάλαιο των νευρωνικών δικτύων κάτι τέτοιο θα σήμαινε τυχαία αρχικοποίηση σε κάποιο διάστημα και σταδιακά θα αλλάζαμε

τα βάρη. Από την στιγμή που οι παράμετροι κάθε μοντέλου δεν είναι μόνον τα βάρη των νευρωνικών δικτύων αλλά και κάποιες παράμετροι που είναι εκεί για την συνέχεια των μοντέλων θα πρέπει να βρούμε και κάποιον τρόπο να τις αρχικοποιήσουμε. Η καλύτερη τιμή που θα μπορούσαν να έχουν αυτές οι μεταβλητές προέρχονται από αποτίμηση της αντικειμενικής συναρτήσεως σε αυτά τα σημεία καθώς και των παραγώγων της, αν αυτό απαιτείται. Ωστόσο κάτι τέτοιο δεν είναι εφικτό για προφανείς λόγους. Από την άλλη διαπιστώθηκε (μετά από πολλά πειράματα) πως η τυχαία αρχικοποίηση αυτών των τιμών δεν οδηγεί σε κάποιο καλό αποτέλεσμα. Για αυτούς τους λόγους επιλέχτηκε πριν από την βασική εκπαίδευση του συστήματος να υπάρξει μία φάση κατά την οποία θα εκτιμηθούν αυτές οι παράμετροι. Αυτή η φάση ονομάστηκε προεπεξεργασία.

Κατά την φάση της προεπεξεργασίας σε κάθε διάστημα εκπαιδεύσεως θεωρούμε πως το μοντέλο προσεγγίσεως είναι ένα απλό νευρωνικό δίκτυο και αγνοούμε προσωρινά την ύπαρξη των εξωτερικών παραμέτρων. Η εκπαίδευση του νευρωνικού δικτύου όπως θα δούμε και στα πειράματα θα γίνεται είτε με την μέθοδο εκπαιδεύσεως TOLMIN είτε με την μέθοδο ευρέσεως καθολικών ελαχίστων CLUSTERING. Μετά το τέλος της προεπεξεργασίας εκτιμούμε σαν αρχική τιμή των εξωτερικών παραμέτρων τις αποτιμήσεις των αντίστοιχων νευρωνικών δικτύων στα αντίστοιχα σημεία. Αυτό έχει δύο βασικές επιπτώσεις στο σύστημά μας:

1. Τα βάρη των νευρωνικών δικτύων δεν χρειάζεται (και δεν πρέπει) να αρχικοποιηθούν, αφού πήραν κάποιες (ίσως καλές) τιμές από το πρώτο στάδιο. Έτσι οι τιμές των βαρών από το στάδιο της προεπεξεργασίας περνούν στο στάδιο της βασικής εκπαιδεύσεως.

2. Σε γειτονικά διαστήματα εκπαιδεύσεως δύο διαφορετικά νευρωνικά δίκτυα θα δίνουν διαφορετικές τιμές για την ίδια εξωτερική παράμετρο. Το σύστημα δεν χειρίζεται αυτό το πρόβλημα, αφού εμείς θέλουμε κάποιες αρχικές τιμές για τις παραμέτρους μας. Εξάλλου σε πολλές περιπτώσεις η προεκπαίδευση μπορεί να δώσει πολύ καλά αποτελέσματα, ώστε να μην χρειασθεί να υπολογίσουμε μέσους όρους.

Το στάδιο της προεπεξεργασίας είναι πλήρως παραλληλοποιήσιμο, αφού δεν απαιτείται από τα νευρωνικά δίκτυα να μοιράζονται κάτι κοινό, ούτε πρότυπα ούτε παραμέτρους.

6.4.4 Βασική Εκπαίδευση

Στην φάση της βασικής εκπαιδεύσεως προσπαθούμε να βρούμε καλύτερες τιμές για τα βάρη του κάθε νευρωνικού δικτύου διατηρώντας σταθερές τις τιμές που έχουμε για τις εξωτερικές παραμέτρους. Αυτό το στάδιο είναι παραλληλοποιήσιμο, υπό την βασική προϋπόθεση πως δεν αλλάζουν οι εξωτερικές παράμετροι. Για την βασική εκπαίδευση επιλέχθηκε η μέθοδος βελτιστοποιήσεως TOLMIN.

6.4.5 Εξομάλυνση

Η φάση της εξομαλύνσεως εκτελείται αμέσως μετά την φάση της βασικής εκπαιδεύσεως. Σε αυτό το στάδιο έχουμε ένα μόνον μοντέλο, το οποίο περιλαμβά-

νει στα πρότυπα εκπαιδεύσεως του όλα τα διαθέσιμα πρότυπα από το αρχείο train.data Αυτό το καθολικό μοντέλο ισούται με τα επιμέρους μοντέλα σε κάθε διάστημα εκπαιδεύσεως. Ωστόσο οι παράμετροι εκπαιδεύσεως (οι μεταβλητές που θα αλλάξουν) δεν είναι τα βάρη των νευρωνικών δικτύων, αλλά οι εξωτερικές παράμετροι. Και σε αυτήν την φάση σαν μέθοδος εκπαιδεύσεως επιλέχθηκε η μέθοδος TOLMIN. Πρέπει να σημειωθεί πως το αρχικό λάθος στην διαδικασία της εξομαλύνσεως ισούται με το άθροισμά των σφαλμάτων που κάνουν τα επιμέρους μοντέλα. Αν για παράδειγμα έχουμε N μοντέλα τότε το αρχικό σφάλμα στην εξομάλυνση θα είναι

$$E = E_0 + E_1 + ... + E_N$$

Μετά το πέρας της εξομαλύνσεως το σφάλμα που θα πάρουμε θα είναι

$$\widetilde{E} = \widetilde{E_0} + \widetilde{E_1} + ... + \widetilde{E_N}$$

Φυσικά θα ισχύει

$$\widetilde{E} \leq E$$

χωρίς αυτό όμως να σημαίνει και αυτόματα πως

$$\widetilde{E_i} \leq E_i, \forall i$$

αφού κατά την διάρκεια της εξομαλύνσεως κάποια επιμέρους λάθη μπορεί να μειωθούν και κάποια άλλα να αυξηθούν. Αυτό δεν πρέπει να μας προβληματίζει, αφού αυτό που ζητάμε είναι η μείωση του συνολικού σφάλματος.

6.4.6 Αλγόριθμος

Το τελευταίο θέμα που έχουμε να δούμε σε αυτό το κεφάλαιο είναι ο αλγόριθμος εκπαιδεύσεως του συστήματος και πιο συγκεκριμένα η βασική συνάρτηση.

Αλγόριθμος Train

1. Ανάγνωση παραμέτρων.
2. Ανάγνωση προτύπων εκπαιδεύσεως.
3. RETRAINS=1
4. Για κάθε διάστημα d κάνε παράλληλα
   (α) N=Νευρωνικό δίκτυο στο d.
   (β) Εκπαίδευση του N (προεπεξεργασία).
5. Αρχικοποίηση των εξωτερικών παραμέτρων.
6. Για κάθε διάστημα d κάνε παράλληλα
   (α) M= Μοντέλο στο d.
   (β) Βασική Εκπαίδευση του M.
7. Εκπαίδευση των εξωτερικών παραμέτρων(εξομάλυνση).
8. E=λάθος εκπαιδεύσεως.
9. Αν E<EPS τότε
   (α) Εκτύπωση αρχείων εξόδου.
   (β) Διακοπή.
10. RETRAINS=RETRAINS+1
11. Αν RETRAINS==MAXRETRAINS τότε
   (α) Εκτύπωση αρχείων εξόδου.
   (β) Διακοπή
12. Πήγαινε στο 6.

Κεφάλαιο 7

Πειράματα

Στο κεφάλαιο αυτό θα δούμε την εφαρμογή του μοντέλου του προηγούμενου κεφαλαίου σε διάφορα προβλήματα στην μία και στις δύο διαστάσεις. Σκοπός των πειραμάτων αυτών είναι να διερευνήσουμε αν το προτεινόμενο μοντέλο είναι χρήσιμο.

7.1 Μία διάσταση

7.1.1 Χαρακτηριστικά πειραμάτων

Χρησιμοποιήθηκαν δύο συναρτήσεις για τα πειράματα. Η πρώτη είναι η $x \sin x^2$, η οποία παρουσιάζει πολύπλοκη δομή και αναμένεται να προσεγγιστεί με κάποια δυσκολία. Η δεύτερη είναι η $e^{\sin x}$ η οποία παρουσιάζει περιοδικότητα. Η γραφική παράσταση της πρώτης στο διάστημα [-4,4] φαίνεται στο Σχήμα 7.1

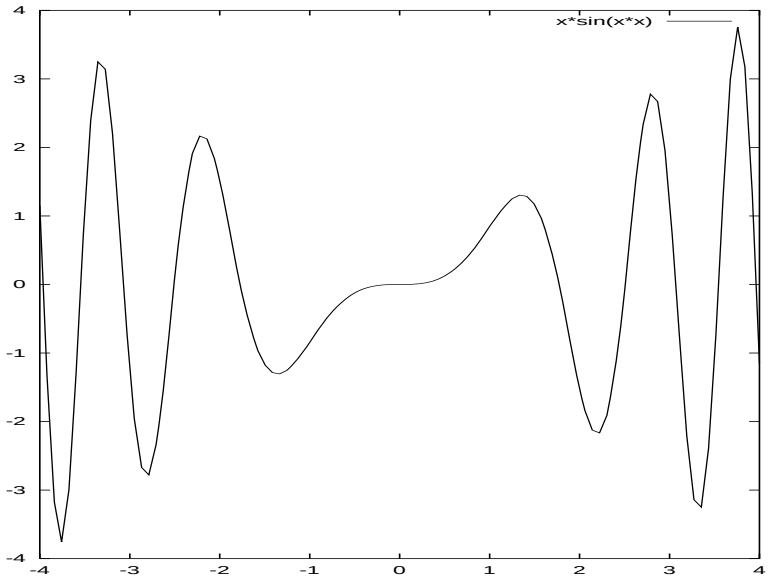
Για την δεύτερη συνάρτηση το γράφημά της στο διάστημα [-10,10] είναι στο Σχήμα 7.2

Τα πειράματα σε κάθε διάσταση χωρίστηκαν σε δύο μεγάλες κατηγορίες. Στην πρώτη κατηγορία επιλέχτηκε να χρησιμοποιηθεί μία μέθοδος τοπικής βελτιστοποιήσεως για την προεπεξεργασία και στην δεύτερη μέθοδο χρησιμοποιήθηκε μία μέθοδος καθολικής βελτιστοποιήσεως.

Η μέθοδος τοπικής βελτιστοποιήσεως που χρησιμοποιήθηκε ήταν η μέθοδος TOLMIN του POWELL, που περιέχεται στο πακέτο βελτιστοποιήσεως MERLIN v3.0.2[3] Ανήκει στην κατηγορία Quasi Newton και χρησιμοποιεί την ενημέρωση BFGS για τον εσσιανό πίνακα.

Η μέθοδος καθολικής βελτιστοποιήσεως που χρησιμοποιήθηκε ήταν η μέθοδος SINGLE LINKAGE CLUSTERING[4]. Για την μονοδιάστατη περίπτωση χρησιμοποιήθηκαν μοντέλα με 1, 2, 4, 8, 10, 15 και 20 διαστήματα. Σε κάθε διάστημα τα νευρωνικά δίκτυα είχαν 1-8 κρυμμένους νευρώνες.

Για την βασική εκπαίδευση του μοντέλου χρησιμοποιήθηκε μόνον η μέθοδος TOLMIN.



Σχήμα 7.1: Γραφική παράσταση της $x \sin x^2$ στο [-4,4]

7.1.2 Προσέγγιση τιμής

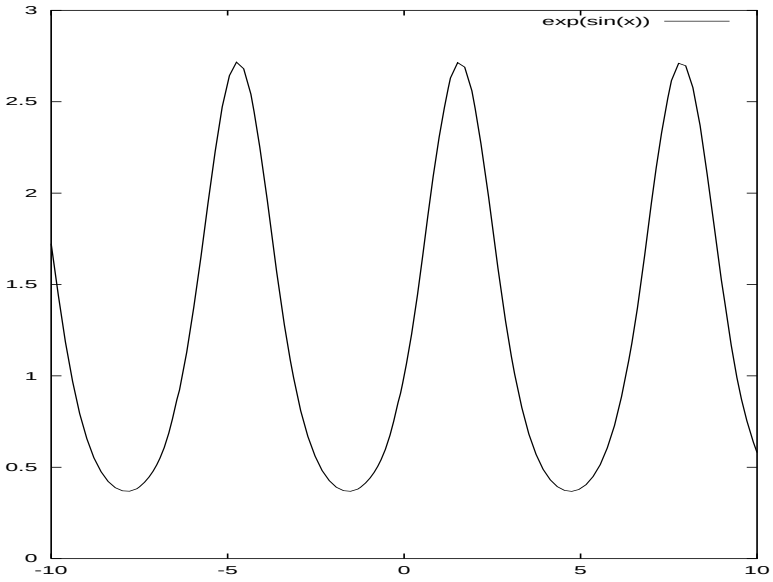
7.1.2.1 Πειράματα με την $x \sin x^2$

Ο τρόπος που επιλέχθηκε να εμφανιστούν τα αποτελέσματα για κάθε προσέγγιση είναι σε κάθε διαμέριση και όχι ανά κρυμμένες μονάδες, γιατί με αυτόν τον τρόπο είναι δυνατόν να κατανοήσουμε κατά πόσον το σύστημα αποδίδει καλύτερα με αύξηση του αριθμού των διαστημάτων.

Συνέχεια συναρτήσεως Παρουσιάζουμε αποτελέσματα για την περίπτωση που επιβάλλουμε συνέχεια της συναρτήσεως μόνον και έχοντας χρησιμοποιήσει τοπική ελαχιστοποίηση στην φάση της προεπεξεργασίας. Στα διαγράμματα που ακολουθούν σε αυτήν την παράγραφο ο κατακόρυφος άξονας αντιστοιχεί στην παράσταση:

$$error(x) = (x \sin x^2 - Model(x))$$

Μπορούμε να κατασκευάσουμε έναν απλό πίνακα με τα μέγιστα και ελάχιστα σφάλματα ανά περίπτωση διαστήματος:

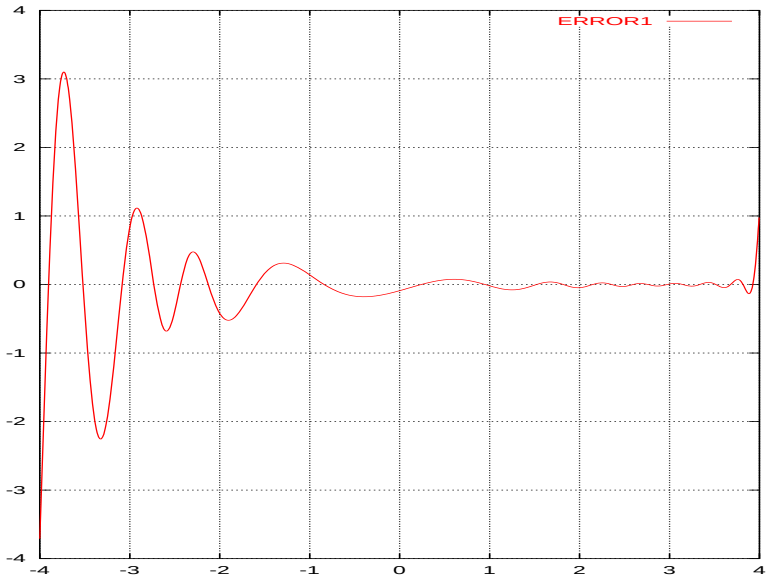


Σχήμα 7.2: Γραφική παράσταση της $e^{\sin x}$ στο [-10,10]

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	477.24	104.11
2	460.02	0.0005
4	389.0	0.0000053
8	14.72	0.106
10	0.77	0.000001
15	0.15	0.0000002
20	0.91	0.0006

Στον πίνακα αυτό και για την συγκεκριμένη συνάρτηση τα μέγιστα σφάλματα προσεγγίσεως τα έχουμε όταν χρησιμοποιούμε έναν κρυμμένο νευρώνα στο σύστημά μας και τα ελάχιστα σφάλματα προσεγγίσεως όταν χρησιμοποιούμε οκτώ νευρώνες. Από το πίνακα μπορούμε να βγάλουμε κάποια συμπεράσματα για την μέθοδό μας. Η αύξηση του αριθμού των διαστημάτων γενικά οδηγεί σε βελτίωση του σφάλματος προσεγγίσεως. Ωστόσο μπορούμε να παρατηρήσουμε πως υπάρχουν περιπτώσεις κατά τις οποίες αυτό δεν συμβαίνει. Αυτό οφείλεται στο ότι στην προεπεξεργασία χρησιμοποιήθηκε τοπική και όχι καθολική τεχνική βελτιστοποιήσεως. Αντίστοιχη είναι η ερμηνεία για την τυχόν αύξηση του σφάλματος προσεγγίσεως αυξανομένων των κόμβων του δικτύου.

Θα ήταν εξαιρετικά χρήσιμο να δούμε πως προσέγγισαν την αντικειμενική συνάρτηση κάποια από τα καλύτερα μοντέλα για μερικές κατηγορίες διαστημάτων. Στο Σχήμα 7.3βλέπουμε την γραφική σύγκριση της αντικειμενικής συναρτήσεως με το καλύτερο μοντέλο του ενός διαστήματος.



Σχήμα 7.3: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου ενός διαστήματος με οκτώ κόμβους για την συνάρτηση $x \sin x^2$

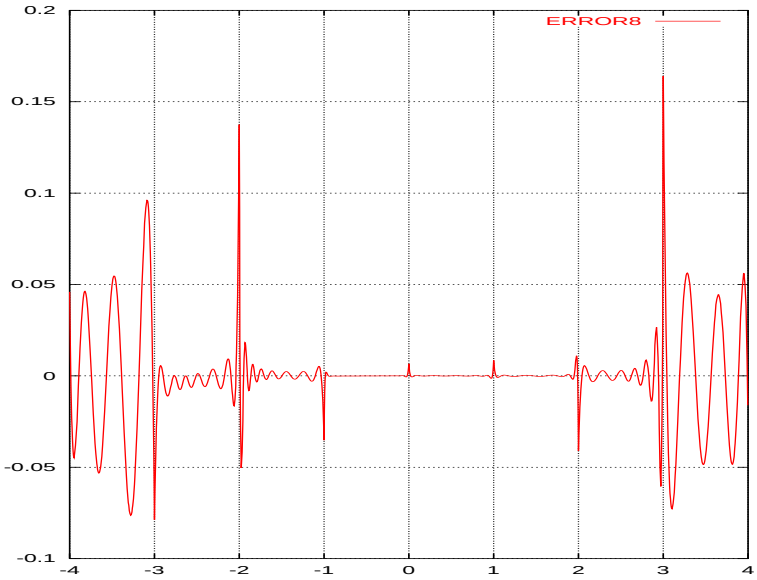
Το μικρότερο σφάλμα προσεγγίσεως στα 8 διαστήματα το είχαμε με 8 κόμβους και δίνεται στο Σχήμα 7.4.

Το μικρότερο σφάλμα προσεγγίσεως από όλες τις περιπτώσεις το είχαμε για 15 διαστήματα με 8 κρυμμένους κόμβους και δίνεται στο Σχήμα 7.5.

Συνέχεια παραγώγου Αν απαιτηθεί και η συνέχεια της πρώτης παραγώγου, τότε καταλήγουμε σε ένα υπολογιστικό μοντέλο με περισσότερες παραμέτρους. Ο πίνακας σφαλμάτων για αυτήν την περίπτωση του μοντέλου μας έχει ως εξής:

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	459.32	113.53
2	424.50	0.00035
4	334.37	0.000001
8	1.31	0.034
10	0.80	0.000009
15	0.87	0.000002
20	0.60	0.052

Επαναλαμβάνουμε τα πειράματά μας με την μόνη διαφορά ότι κατά την φάση της προεπεξεργασίας χρησιμοποιούμε μέθοδο καθολικής ελαχιστοποιήσεως.



Σχήμα 7.4: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου 8 διαστημάτων με οκτώ κόμβους για την συνάρτηση $x \sin x^2$

Συνέχεια Συναρτήσεως Με δεδομένη μόνον την συνέχεια της συναρτήσεως τα συγκριτικά αποτελέσματα που πήραμε εμφανίζονται στον επόμενο πίνακα:

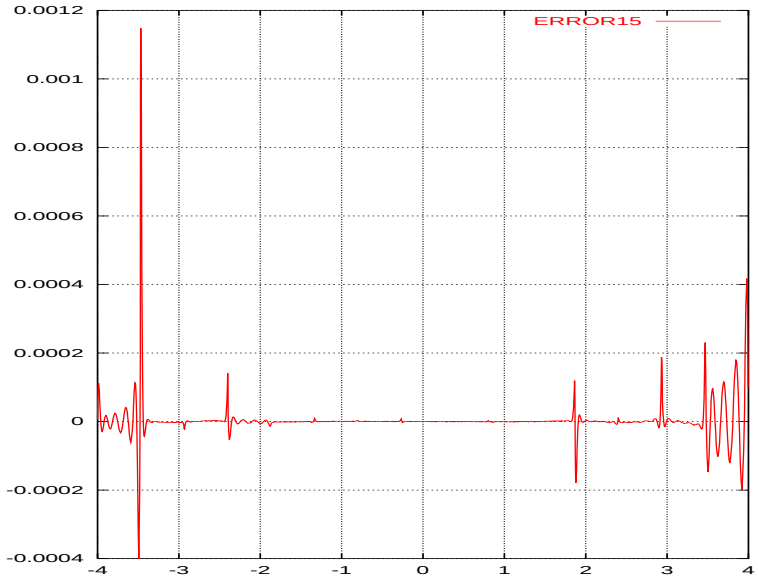
ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	477.24	11.56
2	432.71	0.000011
4	363.31	0.0000003
8	5.40	0.0000002
10	0.75	0.00000002
15	0.09	0.00000001
20	0.90	0.00000001

Από τον παραπάνω πίνακα άμεσα μπορούμε να συμπεράνουμε τα ακόλουθα:

1. Το σφάλμα πέφτει μονότονα με την αύξηση του αριθμού των διαστημάτων.

2. Με την αύξηση του αριθμού των κόμβων στο κρυμμένο επίπεδο το σύστημα βελτιώνει την μαθησιακή του ικανότητα. Αυτό ήταν κάτι που περιμέναμε να συμβεί, αφού έχουμε χρησιμοποιήσει καθολική βελτιστοποίηση

3. Τα σφάλματα είναι πολύ μικρότερα από την περίπτωση που είχαμε την μέθοδο TOLMIN. Επομένως μπορούμε ασφαλώς να συμπεράνουμε πως οι αρχικές συνθήκες του προβλήματος παίζουν μεγάλο ρόλο στην τελική λύση.



Σχήμα 7.5: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου 15 διαστημάτων με οκτώ κόμβους για την συνάρτηση $x \sin x^2$

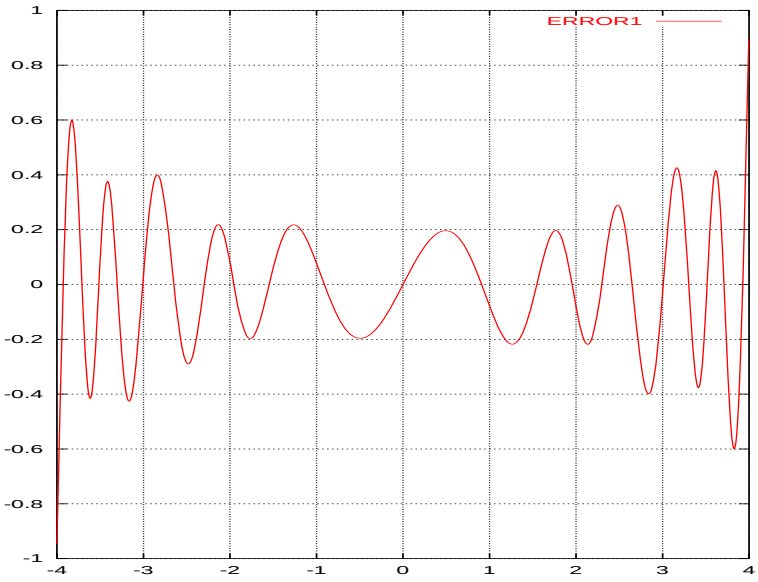
Όπως και με την περίπτωση της μεθόδου TOLMIN, μπορούμε και εδώ να δούμε μερικές γραφικές παραστάσεις σφάλματος προσεγγίσεως για κάποια από τα πειράματα. Για να υπάρχει μία σύγκριση με τα αρχικά πειράματα, θα επιλέξουμε να παρουσιάσουμε τις ίδιες περιπτώσεις στα Σχήματα 7.6, 7.7, 7.8.

Και στις τρεις περιπτώσεις τα σφάλματα προσεγγίσεως έχουν δύο βασικά χαρακτηριστικά:

- Είναι πολύ πιο μικρά από ότι είδαμε στην περίπτωση της προεκπαιδεύσεως με την μέθοδο TOLMIN.

- Είναι περισσότερο κατανεμημένα από ότι προηγουμένως. Αυτό θα πρέπει να το περιμένουμε αφού τα μοντέλα είναι ανεξάρτητα και η αντικειμενική συνάρτηση είναι συμμετρική.

Συνέχεια παραγώγου Αν απαιτήσουμε και συνέχεια παραγώγου, τότε ο αντίστοιχος πίνακας σφάλματος προσεγγίσεως είναι:



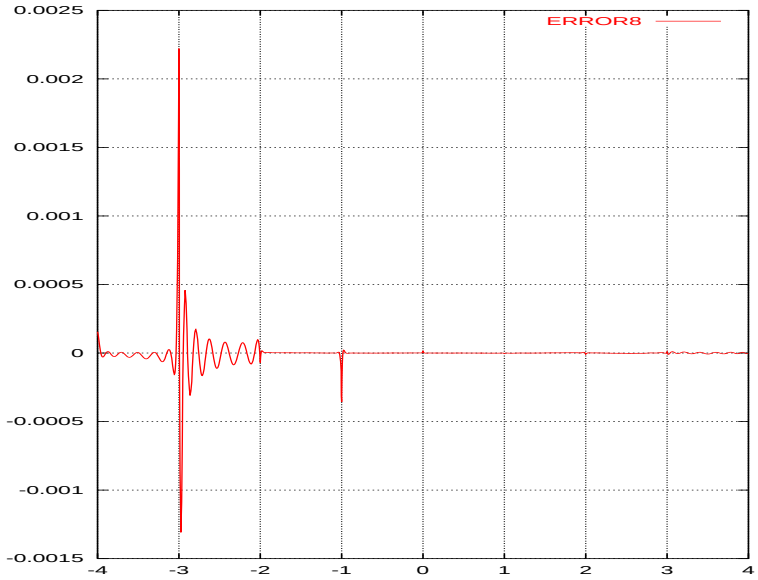
Σχήμα 7.6: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου ενός διαστήματος με οκτώ κόμβους για την συνάρτηση $x \sin x^2$

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	459.32	10.16
2	373.58	0.00002
4	258.96	0.00000005
8	0.79	0.0000009
10	0.86	0.00000005
15	0.86	0.00000001
20	0.65	0.00000001

7.1.2.2 Πειράματα με την $e^{\sin x}$

Η πρώτη συνάρτηση είναι δύσκολο να προσεγγιστεί με λίγα διαστήματα. Από τους πίνακες και τα διαγράμματα που είδαμε μέχρι τώρα ήταν εύκολο να διαπιστώσουμε πως ακόμα και με 4 διαστήματα το πρόβλημα δεν λύνεται, αν δεν διαθέτουμε αρκετούς κόμβους. Από την άλλη το 1 νευρωνικό από μόνο του απαιτεί πολλούς κόμβους για να λύσει το πρόβλημα. Ωστόσο υπάρχουν και προβλήματα τα οποία δεν έχουν πολύπλοκη δομή. Σε αυτά τα προβλήματα θα θέλαμε να διατηρήσουμε τις καλές προσεγγιστικές δυνατότητες ενός απλού μοντέλου. Το δεύτερο πρόβλημα είναι η συνάρτηση $e^{\sin x}$.

Συνέχεια συναρτήσεως Αν επιθυμούμε μόνον συνέχεια συναρτήσεως από το μοντέλο μας και χρησιμοποιήσουμε την μέθοδο TOLMIN στην προεκπαίδευση θα



Σχήμα 7.7: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου οκτώ διαστημάτων με οκτώ κόμβους για την συνάρτηση $x \sin x^2$

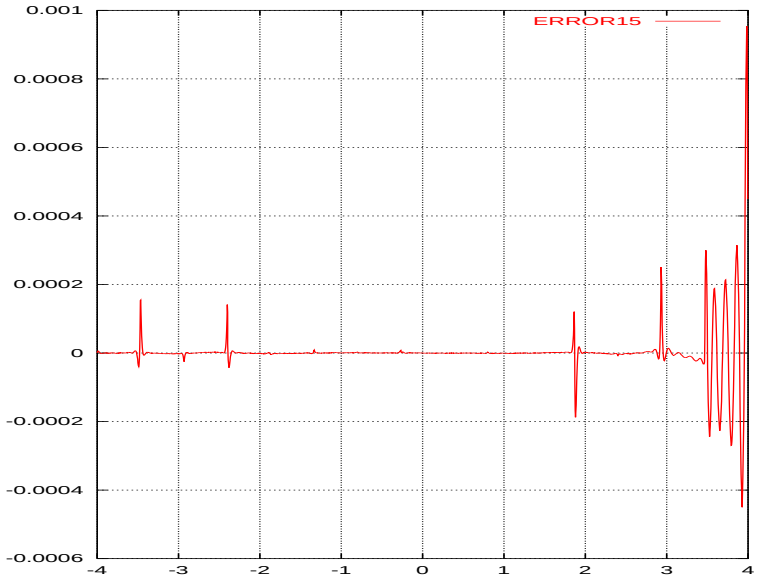
πάρουμε τον επόμενο πίνακα σφαλμάτων προσεγγίσεως:

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	113.58	0.00025
2	65.96	0.00000007
4	29.84	0.00000005
8	0.85	0.00000001
10	0.72	0.00000001
15	0.12	0.00000001
20	0.22	0.00000001

Σχεδόν σε κάθε περίπτωση πετύχαμε ταχεία επίλυση του προβλήματος. Σε πολλές περιπτώσεις μάλιστα δεν χρειάστηκαν πολλές επανεκπαιδεύσεις, αφού το πρόβλημα είχε λυθεί από την φάση της πρώτης επαναλήψεως.

Στην συνέχεια παραθέτουμε τα γραφήματα σφάλματος προσεγγίσεως για τις καλύτερες περιπτώσεις της μεθόδου με 1 διάστημα (Σχήμα 7.9), 8 διαστήματα (Σχήμα 7.10) και 15 διαστήματα (Σχήμα 7.10).

Συνέχεια παραγώγου Αν θεωρήσουμε πως έχουμε μοντέλο με συνέχεια παραγώγων και προεκπαίδευση με χρήση τοπικής βελτιστοποιήσεως παίρνουμε τον επόμενο πίνακα για το σφάλμα προσεγγίσεως:



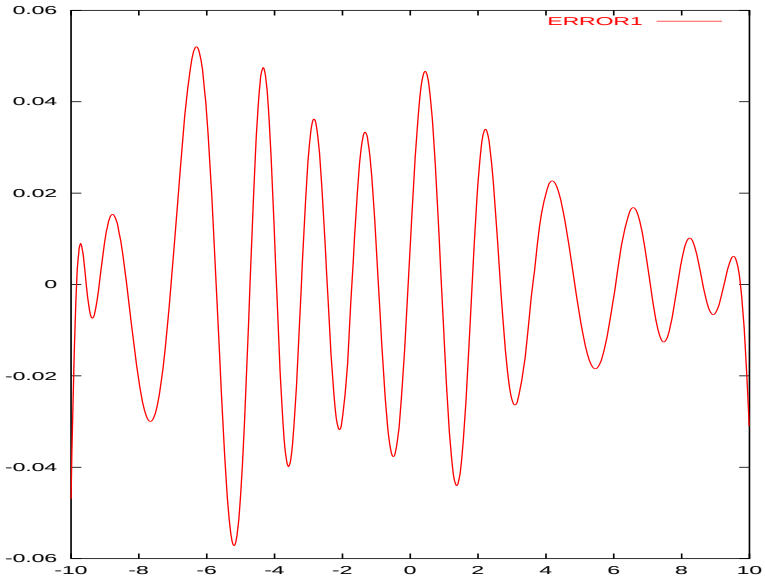
Σχήμα 7.8: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου δεκαπέντε διαστημάτων με οκτώ κόμβους για την συνάρτηση $x \sin x^2$

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	116.37	0.0014
2	71.96	0.00000005
4	4.11	0.00000007
8	0.13	0.00000001
10	0.09	0.00000001
15	0.06	0.00000001
20	0.16	0.00000001

Αυτό που μπορούμε να παρατηρήσουμε είναι πως όπως και πριν είχαμε μεγάλη μείωση στις μέγιστες τιμές σφάλματος προσεγγίσεως που έφτασε μέχρι και το 80%. Μάλιστα αυτήν την φορά είχαμε μείωση στις ελάχιστες τιμές σφάλματος προσεγγίσεως, όπως μπορούμε να δούμε από την τρίτη στήλη του παραπάνω πίνακα.

Στην συνέχεια μένει να δούμε ποια είναι τα αποτελέσματα προσεγγίσεως της παραπάνω συναρτήσεως αν χρησιμοποιήσουμε σαν προεκπαίδευση την μέθοδο καθολικής βελτιστοποιήσεως clustering.

Συνέχεια συναρτήσεως Αν απαιτήσουμε συνέχεια μόνον στις τιμές των μοντέλων ο πίνακας που λαμβάνουμε είναι ο ακόλουθος:

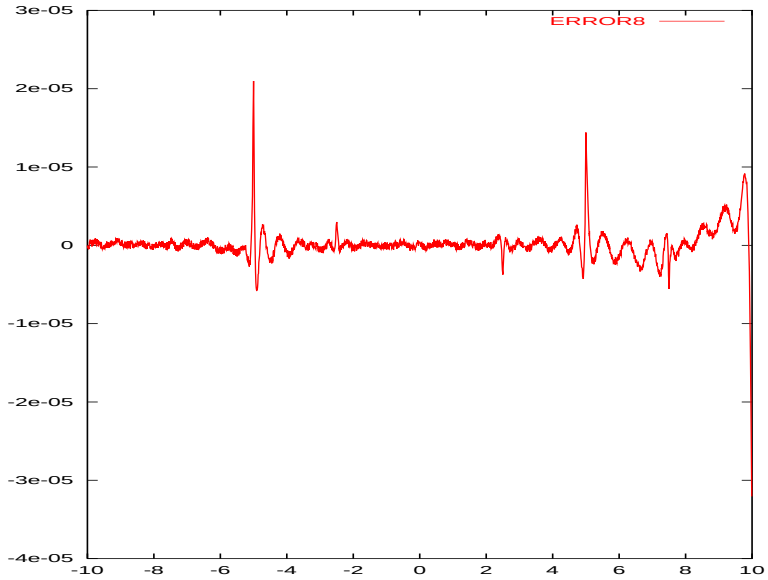


Σχήμα 7.9: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου ενός διαστήματος με οκτώ κόμβους για την συνάρτηση $e^{\sin x}$

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	113.58	0.000007
2	65.96	0.00000007
4	1.50	0.00000002
8	0.84	0.00000001
10	0.11	0.00000001
15	0.06	0.00000001
20	0.04	0.00000001

Δεν μπορούμε να πούμε πως η χρήση της καθολικής βελτιστοποιήσεως ωφέλησε σε μεγάλο βαθμό την προσέγγιση της συναρτήσεως. Τα μέγιστα είναι περίπου ίδια με αυτά που είχαμε στο προηγούμενο σύστημα. Επομένως η χρήση της καθολικής βελτιστοποιήσεως δεν είναι πάντοτε αναγκαία και σε επόμενη ενότητα θα δούμε χρονικά κριτήρια που καθορίζουν πότε είναι συμφέρον να χρησιμοποιούμε αυτήν την τεχνική.

Συνέχεια παραγώγου Αν απαιτήσουμε και συνέχεια παραγώγου από το μοντέλο μας, τότε θα καταλήξουμε στον επόμενο πίνακα:



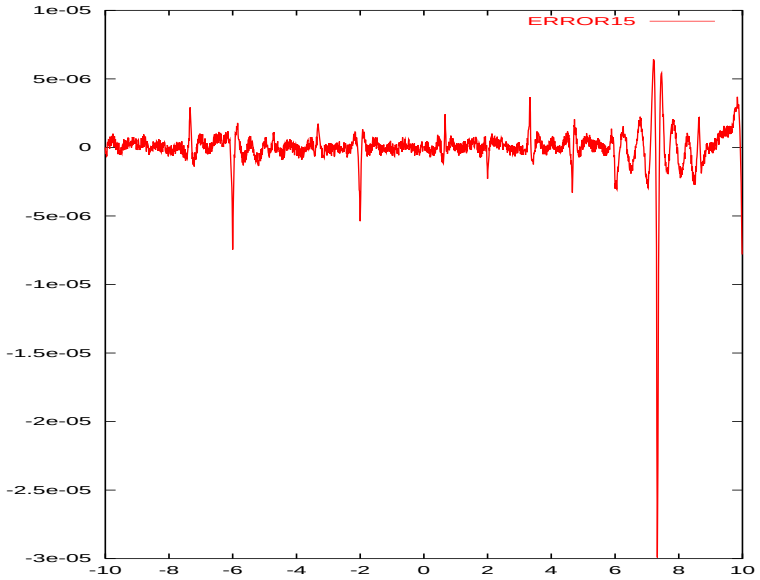
Σχήμα 7.10: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου οκτώ διαστημάτων με οκτώ κόμβους για την συνάρτηση $e^{\sin x}$

ΔΙΑΣΤΗΜΑΤΑ	ΜΕΓΙΣΤΟ	ΕΛΑΧΙΣΤΟ
1	116.37	0.000006
2	71.96	0.00000008
4	0.53	0.00000004
8	0.13	0.00000001
10	0.04	0.00000001
15	0.01	0.00000001
20	0.06	0.00000001

7.1.3 Διαμέριση

Το βασικό ερώτημα που θέτει η μέθοδος είναι αν χρειαζόμαστε την διαμέριση για να επιλύσουμε τα προβλήματα που μας δίνονται. Για να το σκεφτούμε αυτό θα πρέπει να αντιπαραβάλλουμε αυτό που μας δίνει σαν λύση η νέα μέθοδος με ένα μοντέλο στο οποίο δεν έχουμε διαμέριση, για παράδειγμα ένα απλό νευρωνικό δίκτυο. Η απόφασή μας δεν πρέπει να βασιστεί μόνον στο σφάλμα προσεγγίσεως αλλά και στον χρόνο που επιτυγχάνεται κάτι τέτοιο. Το γράφημα του Σχήματος 7.12 παριστάνει το σφάλμα προσεγγίσεως που έχουμε με την χρήση ενός νευρωνικού για την συνάρτηση $x \sin x^2$. Το νευρωνικό δίκτυο υλοποιήθηκε στο σύστημα βελτιστοποιήσεως Merlin 3.02 και εκπαιδεύτηκε με την βοήθεια της μεθόδου Levenberg - Marquardt.

Το σφάλμα προσεγγίσεως είναι αρκετά μεγάλο, ακόμα και όταν έχουμε με-



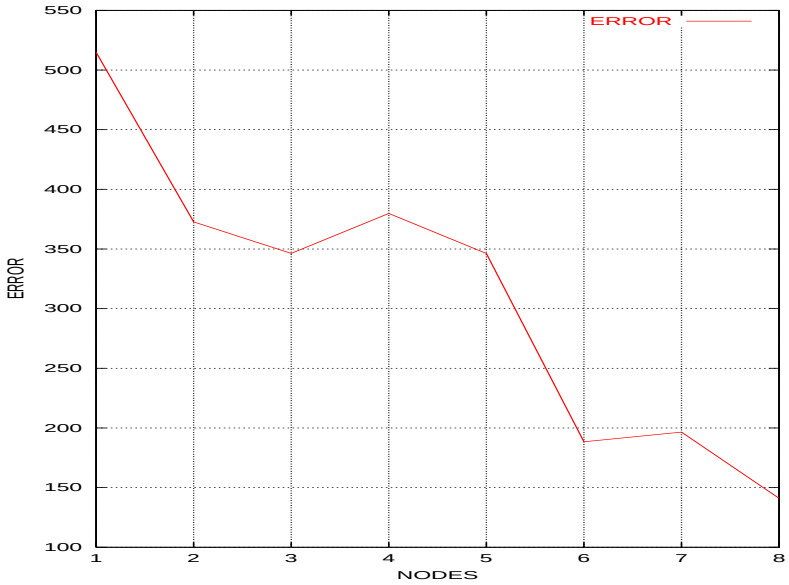
Σχήμα 7.11: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου δεκαπέντε διαστημάτων με οκτώ κόμβους για την συνάρτηση $e^{\sin x}$

γάλο αριθμό κόμβων. Από την άλλη αν κοιτάξουμε στο Σχήμα 7.13τα χειρότερα των σφαλμάτων προσεγγίσεως που γίνονται από το μοντέλο μας

Θα διαπιστώσουμε πως από την διαμέριση με οκτώ διαστήματα και έπειτα το **μέγιστο** λάθος σχεδόν μηδενίζεται. Το μέγιστο σφάλμα προσεγγίσεως το έχουμε στις παραπάνω περιπτώσεις με έναν μόνο κρυμμένο νευρώνα σε κάθε επιμέρους νευρωνικό δίκτυο. Αν τώρα θελήσουμε να δούμε και ποια είναι τα καλύτερα σφάλματα στην προσέγγιση που επιτυγχάνουμε θα πάρουμε τον επόμενο πίνακα

ΔΙΑΣΤΗΜΑΤΑ	ΣΦΑΛΜΑ
1	104.11
2	0.0005
4	0.0000053
8	0.106
10	0.000001
15	0.0000002
20	0.0006

Τα παραπάνω σφάλματα στην προσέγγιση επιτεύχθηκαν με οκτώ κρυμμένους κόμβους ανά δίκτυο. Ακόμα και στην περίπτωση του ενός δικτύου είχαμε καλύτερο αποτέλεσμα με την νέα μέθοδο. Ωστόσο για να κρίνουμε την αναγκαιότητα της νέας μεθόδου θα πρέπει να εξετάσουμε και τον χρόνο που χρειαζόμαστε για την εκτέλεση των πειραμάτων. Στον επόμενο πίνακα έχουμε τους χρόνους που χρειάστηκε το ένα νευρωνικό για την όποια επίλυση του προβλήματος:

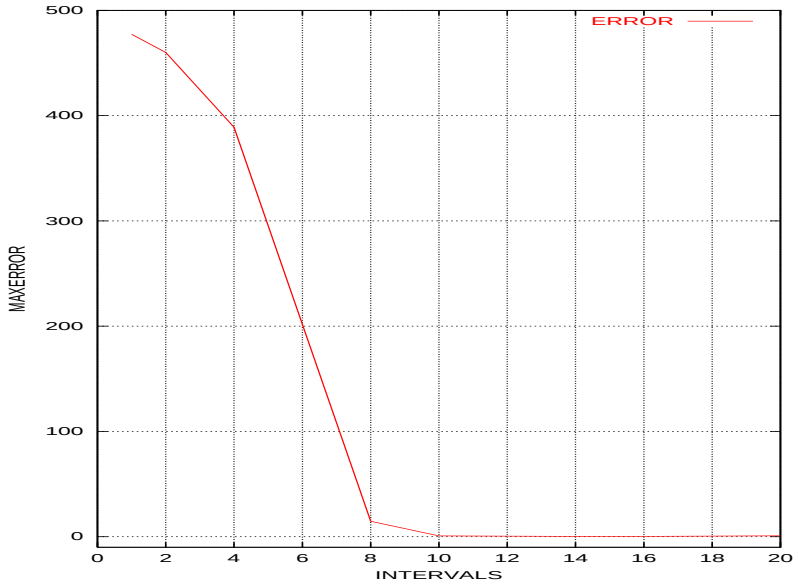


Σχήμα 7.12: Σφάλμα προσεγγίσεως ενός νευρωνικού δικτύου για την συνάρτηση $x \sin x^2$

ΚΟΜΒΟΙ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ
1	514.95	4.257
2	372.71	5.892
3	346.33	7.698
4	379.76	16.977
5	346.23	8.175
6	188.42	28.306
7	196.50	34.585
8	141.04	234.173

Γενικά ο χρόνος δεν παίρνει μεγάλες τιμές. Ωστόσο το πρόβλημα δεν λύνεται. Στις καλύτερες περιπτώσεις επιλύσεως από το μοντέλο μας οι αντίστοιχοι χρόνοι ήταν:

ΔΙΑΣΤΗΜΑΤΑ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ
1	104.11	353.97
2	0.00052	997.35
4	0.00000053	994.34
8	0.106	813.07
10	0.00000582	1315.50
15	0.00000019	199.11
20	0.00596	815.61



Σχήμα 7.13: Μέγιστα σφάλματα προσεγγίσεως προτεινόμενου μοντέλου σε κάθε διαμέριση για την συνάρτηση $x \sin x^2$

Οι χρόνοι είναι μεγαλύτεροι από την περίπτωση ενός νευρωνικού δικτύου ωστόσο πρέπει να προσέξουμε τα επόμενα:

1. Το μοντέλο στις περισσότερες περιπτώσεις είχε ικανοποιητικά σφάλματα στην προσέγγιση.

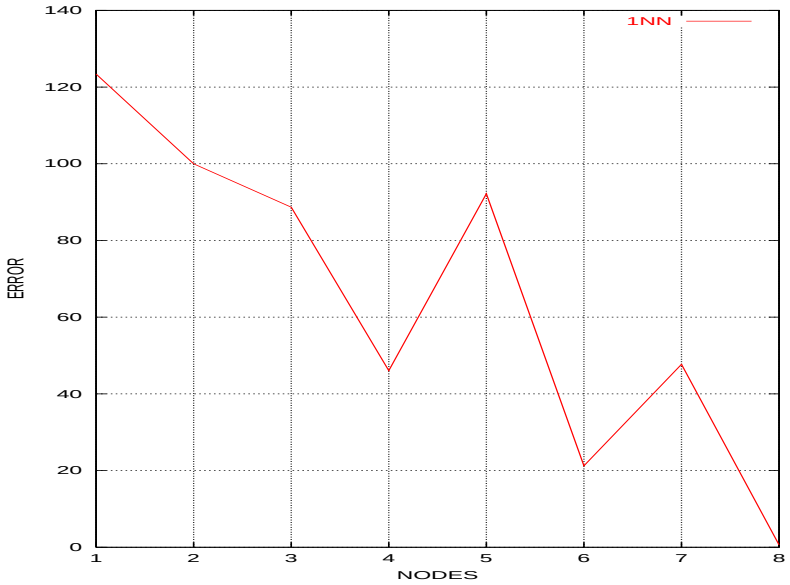
2. Το πλήθος των παραμέτρων εκπαιδεύσεως είναι γενικά περισσότερο στο νέο μοντέλο.

3. Στο νέο μοντέλο ένα σημαντικό μέρος του χρόνου χάνεται στην ανταλλαγή μηνυμάτων στα μηχανήματα που συνθέτουν την ομάδα εργασίας της MPI.

4. Η αύξηση του πλήθους διαστημάτων δεν συνεπάγεται άμεσα και αύξηση του χρόνου.

5. Η αύξηση των διαστημάτων γενικά οδηγεί σε διαμέριση του προβλήματος σε ευκολότερα προβλήματα σε κάθε περιοχή. Αυτό σημαίνει πως τα επιμέρους προβλήματα επιλύονται ταχύτερα.

Η τελευταία διαπίστωση από τις παραπάνω είναι περισσότερο φανερή αν χρησιμοποιήσουμε την μέθοδο CLUSTERING για προεκπαίδευση του συστήματός μας. Σε αυτήν την περίπτωση τα καλύτερα σφάλματα προσεγγίσεως που επιτυγχάνονται σχετίζονται με τον χρόνο σύμφωνα με τον επόμενο πίνακα:



Σχήμα 7.14: Σφάλμα προσεγγίσεως ενός νευρωνικού δικτύου για την συνάρτηση $e^{\sin x}$

ΔΙΑΣΤΗΜΑΤΑ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ
1	11.556	144.47
2	0.00001	296.53
4	0.00000003	126.53
8	0.00000019	103.76
10	0.00000002	88.40
15	0.000000001	279.67
20	0.000000001	534.49

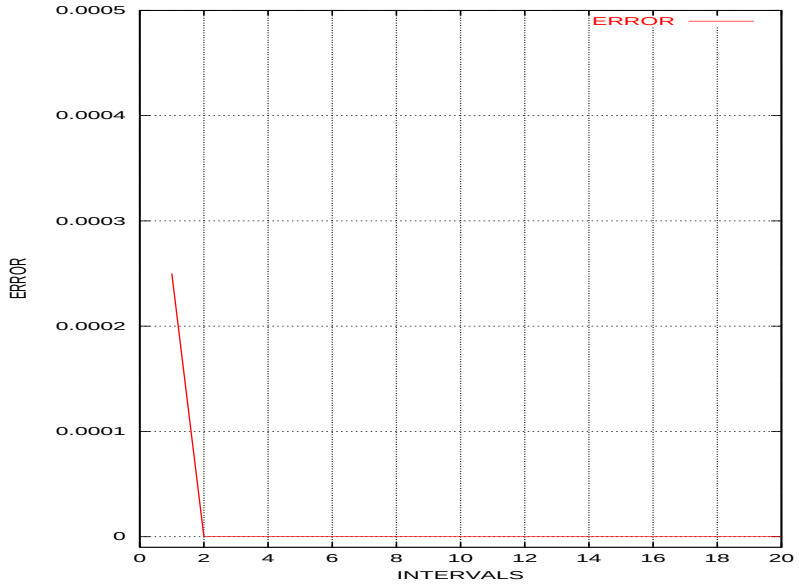
Για να επιτύχουμε ικανοποιητική λύση δεν χρειάστηκε αρκετός χρόνος. Στις περισσότερες περιπτώσεις ο χρόνος ήταν αντίστοιχος ή μικρότερος της περιπτώσεως ενός νευρωνικού δικτύου. Συνεπώς μπορούμε να καταλήξουμε με ασφάλεια στο συμπέρασμα πως η νέα μέθοδος

1. Επιλύει με ευκολία δύσκολα προβλήματα.

2. Δεν απαιτείται αρκετός χρόνος για την επίλυση των προβλημάτων.

Ωστόσο μπορούμε να δοκιμάσουμε να εξάγουμε κάποια συμπεράσματα και για μία πιο εύκολη συνάρτηση όπως είναι η $e^{\sin x}$. Το γράφημα λαθών με ένα μόνον νευρωνικό δίκτυο δίνεται στο Σχήμα 7.14.

Βλέπουμε πως αν και έχουμε μικρότερες τιμές για το σφάλμα προσεγγίσεως από ότι προηγουμένως, οι απόλυτες τιμές παραμένουν υψηλές εκτός και αν δια-



Σχήμα 7.15: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου για κάθε διαμέριση για την συνάρτηση $e^{\sin x}$

θέσουμε αρκετούς κόμβους στο δίκτυό μας. Αν χρησιμοποιήσουμε τις καλύτερες επιδόσεις του μοντέλου μας θα πάρουμε το c γράφημα του Σχήματος 7.15.

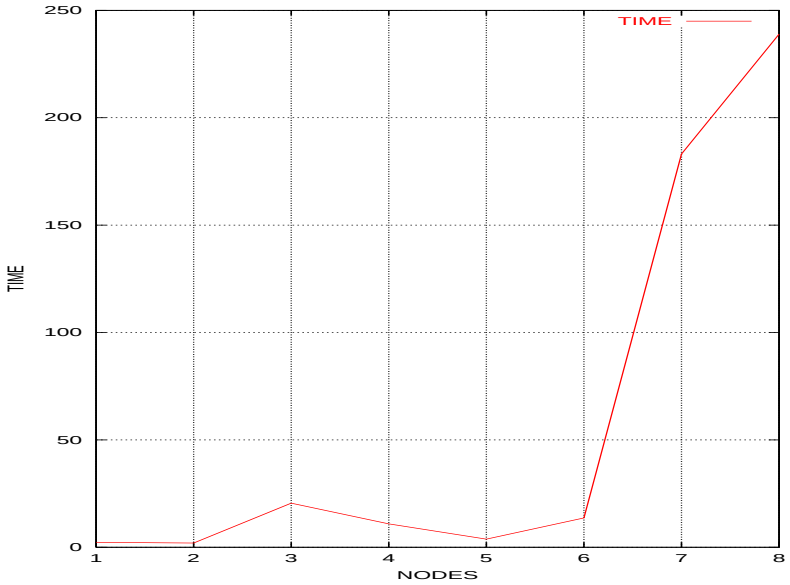
Το σύστημα ακόμα προσέγγισε την συνάρτηση με χαρακτηριστική άνεση ακόμα και με λίγα διαστήματα. Μάλιστα η προσθήκη διαστημάτων πέρα από το 2 δεν δείχνει να προσφέρει κάτι ουσιαστικό. Άρα και με λίγα διαστήματα μπορούμε να προσεγγίσουμε την λύση, μειώνοντας έτσι τον απαιτούμενο αριθμό παραμέτρων. Αν θέλουμε να δούμε τον χρόνο για την λύση που προσέφερε το πρώτο μοντέλο παίρνουμε το γράφημα χρόνου - κόμβων του Σχήματος 7.16.

Από την άλλη αν θεωρήσουμε την περίπτωση με τέσσερα διαστήματα (που σχεδόν έχει λύσει το πρόβλημα) το γράφημα χρόνου - διαστημάτων δίνεται στο Σχήμα 7.17.

Το σύστημα μπόρεσε να λύσει το πρόβλημα σε χρόνο μικρότερο ακόμα και από αυτόν του ενός μοντέλου. Μάλιστα αν αυξήσουμε το πλήθος των μοντέλων θα δούμε πως σε πολλές περιπτώσεις ο χρόνος μειώνεται δραστικά, αφού κάθε επιμέρους μοντέλο λύνει ένα σχετικά εύκολο πρόβλημα.

7.1.4 Καθολική βελτιστοποίηση

Το δεύτερο ερώτημα που καλούμαστε να απαντήσουμε είναι η αναγκαιότητα της καθολικής βελτιστοποιήσεως. Θα πρέπει να επισημανθεί πως η καθολική βελτιστοποίηση δεν γίνεται σε κάθε βήμα του αλγορίθμου, αλλά μόνον κατά την αρχικοποίηση του συστήματος. Για να αποκτήσουμε μία καλύτερη εικόνα της διαφοράς των δύο μεθόδων αρχικοποιήσεως στους επόμενους δύο πίνακες συγκρίνουμε τις



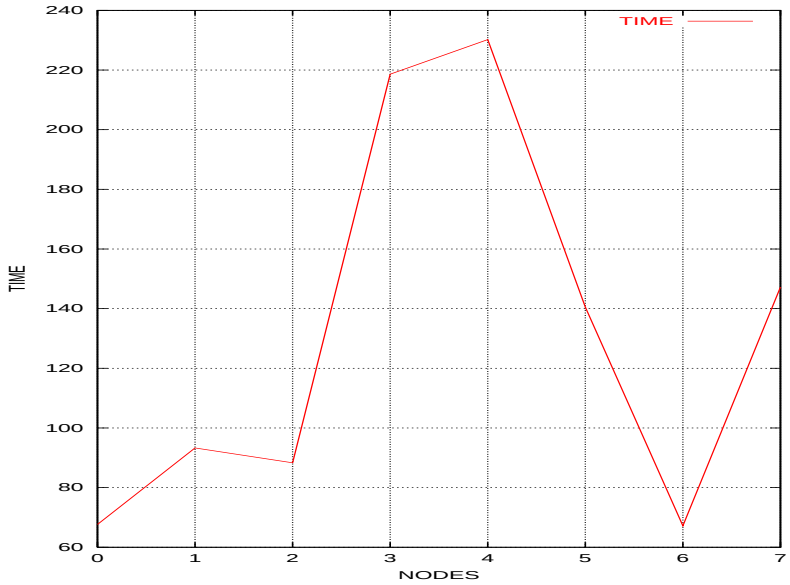
Σχήμα 7.16: Ο χρόνος ενός νευρωνικού δικτύου σε συνάρτηση με τους κόμβους για την προσέγγιση της συναρτήσεως $x \sin x^2$

καλύτερες και τις χειρότερες επιδόσεις των δύο τεχνικών για το πρόβλημα της προσεγγίσεως του $x \sin x^2$.

Αν δούμε τις καλύτερες επιδόσεις των δύο μεθόδων προεκπαιδεύσεως θα πάρουμε τον επόμενο πίνακα:

ΔΙΑΣΤΗΜΑΤΑ	TOLMIN	CLUSTERING	ΔΙΑΦΟΡΑ(%)
1	104.11	11.56	-88.90%
2	0.0005	0.000011	-97.80%
4	0.0000053	0.0000003	-94.34%
8	0.106	0.0000002	-99.99%
10	0.000001	0.00000002	-98.00%
15	0.0000002	0.00000001	-95.00%
20	0.0006	0.00000005	-99.99%

Όπως παρατηρεί κανείς η χρήση της μεθόδου καθολικής βελτιστοποιήσεως έλυσε το πρόβλημα προσεγγίσεως με 2 ή και με 4 διαστήματα. Επίσης οι μεταβολές στα λάθη σχεδόν πάντα ξεπερνούν το 90%. Όμως για να σχηματίσουμε μία πλήρη εικόνα για την αναγκαιότητα της CLUSTERING θα πρέπει να δούμε και τους χρόνους που χρειαστήκαμε για τα παραπάνω εντυπωσιακά αποτελέσματα.



Σχήμα 7.17: Χρόνος προτεινόμενου μοντέλου τεσσάρων διαστημάτων σε συνάρτηση με τους κόμβους για την προσέγγιση της συναρτήσεως $x \sin x^2$

ΔΙΑΣΤΗΜΑΤΑ	TOLMIN	CLUSTERING	ΔΙΑΦΟΡΑ(%)
1	353.97	144.47	-59.20%
2	997.35	296.53	-70.27%
4	994.34	126.53	-87.01%
8	813.07	103.76	-87.24%
10	1152.69	88.40	-92.33%
15	199.11	276.67	+39.01%
20	815.61	534.49	-34.47%

Όταν το σύστημά μας επιτυγχάνει τους στόχους του, τότε ο χρόνος μειώνεται αντί να αυξάνει, αφού η εκπαίδευση των νευρωνικών δικτύων διακόπτεται. Επομένως μπορούμε να πούμε πως η χρήση τεχνικών καθολικής βελτιστοποιήσεως είναι απαραίτητη προκειμένου να έχουμε καλά αποτελέσματα προσεγγίσεως. Ανάλογα αποτελέσματα με τα παραπάνω μπορούμε να παρατηρήσουμε και για την συνάρτηση $e^{\sin x}$.

7.1.5 Επιλογή μεθόδου

Το επόμενο θέμα που έχουμε να δούμε είναι η αναγκαιότητα της μεθόδου με συνέχεια παραγώγων. Θα επικεντρώσουμε την μελέτη μας στην συνάρτηση $x \sin x^2$, καθώς παρουσιάζει όπως είδαμε αρκετά περίπλοκη δομή. Στους πίνακες που ακολουθούν θα χρησιμοποιήσουμε μόνον τα αποτελέσματα που πήραμε με την βοήθεια της CLUSTERING και όχι με αυτά που πήραμε από την TOLMIN, καθώς πρέπει

να είναι φανερό ήδη από τα προηγούμενα πως η προεκπαίδευση με CLUSTERING βελτιώνει σημαντικά την απόδοση του συστήματος. Επίσης θα εξετάσουμε μόνον τις περιπτώσεις κατά τις οποίες είχαμε ικανοποιητικά αποτελέσματα σφάλματος προσεγγίσεως.

Για την μέθοδο με συνέχεια τιμών ο πίνακας σφάλματος προσεγγίσεως και χρόνου έχει ως εξής:

ΔΙΑΣΤΗΜΑΤΑ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ
1	11.56	144.47
2	0.000011	296.53
4	0.0000003	126.53
8	0.0000002	103.76
10	0.00000002	88.40
15	0.00000001	276.67
20	0.00000001	534.49

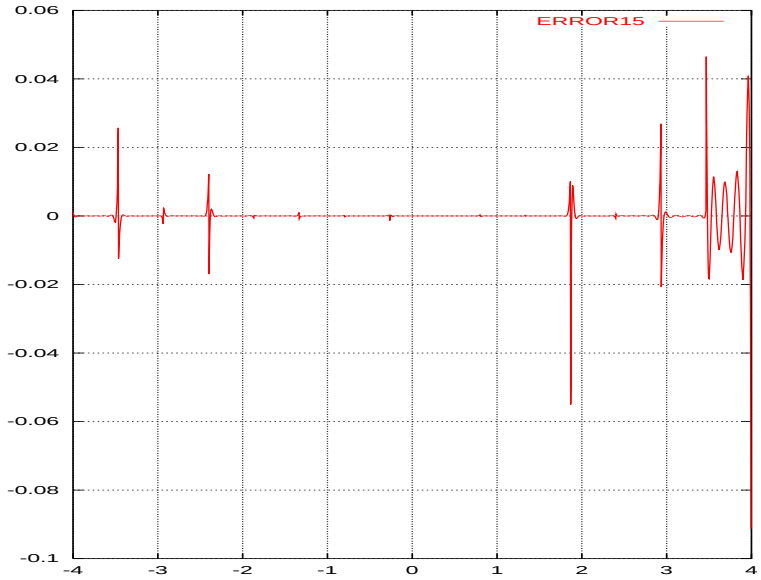
Για την μέθοδο με συνέχεια παραγώγων ο αντίστοιχος πίνακας είναι:

ΔΙΑΣΤΗΜΑΤΑ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ
1	10.16	257.47
2	0.00002	1028.73
4	0.00000005	216.68
8	0.000005	657.50
10	0.00000005	293.98
15	0.00000006	415.31
20	0.00000007	1308.53

Συγκρίνοντας τα καλύτερα σφάλματα προσεγγίσεως των δύο τεχνικών παίρνουμε τον επόμενο πίνακα :

ΔΙΑΣΤΗΜΑΤΑ	ΔΙΑΦΟΡΑ(%)
1	-12.11%
2	+81.18%
4	-83.33%
8	+24.00%
10	+150.00%
15	+500.00%
20	+600%

Στην περίπτωση που έχουμε λίγα διαστήματα μπορούμε να πούμε πως η μέθοδος με συνέχεια παραγώγων επιτυγχάνει καλύτερα αποτελέσματα. Μάλιστα αν μελετήσουμε τα αποτελέσματα που πήραμε και ως προς τον αριθμό των κόμβων, θα διαπιστώσουμε πως η μέθοδος 1 μικρότερα σφάλματα προσεγγίσεως στις χειρότερες περιπτώσεις. Ωστόσο αυτό που μας ενδιαφέρει είναι η επιτυχία της μεθόδου όταν το πρόβλημα λύνεται. Όπως δείχνει ο προηγούμενος πίνακας σε τέτοιες περιπτώσεις η μέθοδος 1 δεν είναι καλύτερη από την μέθοδο 0. Ωστόσο το



Σχήμα 7.18: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου 15 διαστημάτων με 8 κόμβους για την συνάρτηση $x \sin x^2$

μεγάλο πλεονέκτημα της μεθόδου 1 είναι πως εξασφαλίζει συνέχεια παραγώγων. Όμως αν πάρουμε τις διαφορές των χρόνων:

ΔΙΑΣΤΗΜΑΤΑ	ΔΙΑΦΟΡΑ (%)
1	+78.22%
2	+246.92%
4	+71.25%
8	+533.67%
10	+232.56%
15	+50.10%
20	+144.82%

Αυτό που παρατηρούμε είναι πως ο χρόνος αυξάνεται πάρα πολύ στην μέθοδο 1 . Η αύξηση στον χρόνο και η χειρότερη πορεία στο σφάλμα προσεγγίσεως αρκούν για να μην χρησιμοποιηθεί η μέθοδος 1, αφού η συνέχεια παραγώγων δεν είναι αρκετή. Στο Σχήμα 7.18έχουμε το σφάλμα προσεγγίσεως παραγώγου για την μέθοδο 0 με 15 διαστήματα και 8 παραγώγους.

Το σφάλμα προσεγγίσεως είναι αρκετά μικρό, ώστε να είναι προτιμότερη η καλύτερη προσέγγιση της τιμής από την επιβολή συνέχειας στην παράγωγο.

7.1.6 Πολλοί επεξεργαστές

Η επιλογή της χρήσεως πολλών επεξεργαστών έγινε σίγουρα για να μειωθεί ο χρόνος που απαιτείται για την προσέγγιση των δεδομένων συναρτήσεων. Θα μπορούσαμε να προσομοιώσουμε τους παράλληλους επεξεργαστές χρησιμοποιώντας μόνον ένα επεξεργαστή και εκτελώντας την εκπαίδευση σε κάθε διάστημα στην σειρά. Στον επόμενο πίνακα παρουσιάζεται η σύγκριση χρονικής διάρκειας όταν εκτελούμε το πείραμα προσεγγίσεως για την συνάρτηση $x \sin x^2$ σε έναν επεξεργαστή και σε περισσότερους:

Δ	ΚΟΜΒΟΙ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ-1	ΧΡΟΝΟΣ-Ν	ΔΙΑΦΟΡΑ(%)
2	8	0.0021	603.19	296.53	-50.84%
4	8	0.00012	476.90	126.53	-73.47%
8	4	0.0038	857.46	182.86	-78.67%
10	2	0.072	620.96	135.71	-78.15%
10	4	0.0054	866.38	193.55	-77.66%
15	1	0.073	1065.74	216.87	-79.65%
15	2	0.015	1187.66	282.51	-76.21%
20	2	0.054	1717.49	499.90	-70.89%

Στην πρώτη στήλη έχουμε το πλήθος των διαστημάτων που χρησιμοποιήθηκαν και στην δεύτερη το πλήθος των κόμβων επεξεργασίας. Στην τρίτη στήλη έχουμε το μέγιστο απόλυτο σφάλμα προσεγγίσεως που παρατηρήθηκε το οποίο κυμαίνεται ανάμεσα στο 0.054 και το 0.0001. Σημειώνεται πως το σφάλμα αυτό μετρήθηκε μόνον με την μέθοδο CLUSTERING, καθώς όπως παρατηρήθηκε προηγουμένως προσφέρει καλύτερες προσεγγιστικές ικανότητες. Στην τέταρτη στήλη έχουμε τον χρόνο που χρειάστηκε ο ένας επεξεργαστής για την εκτέλεση του πειράματος και στην πέμπτη στήλη έχουμε τον χρόνο που χρειάστηκαν οι Ν επεξεργαστές. Στην μονοδιάστατη και μόνο περίπτωση το Ν αυτό ισούται με το πλήθος των διαστημάτων. Στην τελευταία στήλη έχουμε το ποσοστό χρόνου που κερδίσαμε με την χρήση επιπλέον επεξεργαστών. Από τα παραπάνω αποτελέσματα μπορούμε να εξάγουμε τα ακόλουθα συμπεράσματα:

1. Το γινόμενο ΔΙΑΣΤΗΜΑΤΑ * ΚΟΜΒΟΙ κυμαίνεται ανάμεσα στο 15 και το 40. Υπάρχει διαχωρισμός της επιδόσεως σε δύο κατηγορίες: μία γύρω από το γινόμενο 15 και μία γύρω από το 40 με την δεύτερη να υπερτερεί σε απόδοση.

2. Με την χρήση επιπλέον επεξεργαστών κερδίζουμε από 50 μέχρι και 80% του χρόνου που θα κάναμε με έναν επεξεργαστή. Αυτό σημαίνει πως στον χρόνο του ενός σειριακού πειράματος θα μπορούσαμε να έχουμε μέχρι και 5 παράλληλα.

3. Καθώς προσθέτουμε επεξεργαστές το ποσοστό κέρδους αυξάνει ανάλογα. Μάλιστα από ένα σημείο και έπειτα εμφανίζει αυξητικές τάσεις. Αυτό οφείλεται σε δύο μεγάλους παράγοντες που επηρρεάζουν το σύστημα βελτιστοποιήσεως:

(α) Υπάρχει απώλεια στον χρόνο εξαιτίας της απαιτήσεως για επικοινωνία ανάμεσα στους εμπλεκόμενους επεξεργαστές. Αυτή η απώλεια είναι μεγαλύτερη όταν οι επεξεργαστές είναι απομακρυσμένοι δικτυακά και όταν υπάρχει μεγάλος φόρτος στις γραμμές του δικτύου.

(β) Μετά την εκπαίδευση των ΤΝΔ σε κάθε απομακρυσμένο επεξεργαστή έχουμε την φάση της εξομαλύνσεως (εξωτερικής εκπαιδεύσεως) που εκτελείται σε έναν επεξεργαστή. Αυτός ο χρόνος είναι ίδιος και στην παράλληλη και την σειριακή περίπτωση. Μάλιστα όσο αυξάνεται το πλήθος των διαστημάτων αυτός ο χρόνος τείνει να αυξηθεί, προκαλώντας φυσικά πτώση στο κέρδος που έχουμε από την χρήση πολλών επεξεργαστών.

7.1.7 Πλήθος κόμβων

Το τελευταίο θέμα που θα εξετάσουμε στα μονοδιάστατα πειράματα είναι η συνάρτηση του σφάλματος προσεγγίσεως σε σχέση με τον αριθμό των κόμβων. Μέχρι τώρα μελετήσαμε τις ακραίες περιπτώσεις των διαμορφώσεων των νευρωνικών δικτύων και σε αυτήν την υποενότητα θα δούμε πως σχετίζεται το σφάλματος προσεγγίσεως με τις ενδιάμεσες περιπτώσεις των διαστημάτων. Θα επικεντρώσουμε την μελέτη μας στην συνάρτηση $x \sin x^2$ με προεκπαίδευση TOLMIN και CLUSTERING και μέθοδο 0. Στους επόμενους 7 πίνακες παρουσιάζουμε την ποσοστιαία πτώση του σφάλματος προσεγγίσεως για 1, 2, 4, 8, 10, 15, 20 διαστήματα καθώς το πλήθος των κόμβων αυξάνει από το ένα προς το οκτώ. Η πρώτη στήλη είναι το πλήθος των κόμβων, η δεύτερη στήλη είναι η μεταβολή του σφάλματος προσεγγίσεως για προεκπαίδευση TOLMIN και η τρίτη στήλη για την μεταβολή του σφάλματος προσεγγίσεως με προεκπαίδευση CLUSTERING. Ο πίνακας για το σφάλμα προσεγγίσεως ενός διαστήματος έχει ως ακολούθως:

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	-9.50%	-9.33%
3	-15.39%	-24.63%
4	+4.73%	-13.00%
5	-31.85%	-9.84%
6	-48.51%	-54.37%
7	+58.70%	-53.44%
8	-51.27%	-78.73%

Στην τεχνική CLUSTERING έχουμε διαρκή πρώτη του σφάλματος προσεγγίσεως και σε πολλές περιπτώσεις με υψηλούς ρυθμούς. Αντίθετα με την χρήση TOLMIN σε αρκετές περιπτώσεις είχαμε άνοδο του λάθους, κάτι που οφείλεται στην τυχαιότητα της αρχικοποιήσεως των νευρωνικών δικτύων, αλλά και στο ότι τα νευρωνικά δίκτυα παγιδεύονται σε τοπικά ελάχιστα. Αν χρησιμοποιήσουμε δύο διαστήματα ο πίνακας μεταβολής του σφάλματος προσεγγίσεως έχει ως εξής:

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	-14.03%	-40.36%
3	+64.41%	-88.33%
4	-3.78%	-84.29%
5	-57.25%	-80.94%
6	+0.90%	-84.26%
7	-99.98%	-99.77%
8	-98.87%	-99.67%

Το ίδιο φαινόμενο που παρατηρήσαμε στην προσέγγιση με ένα νευρωνικό παρατηρείται και εδώ, μόνον που σε διαμέριση δύο τμημάτων η ταχύτητα συγκλήσεως αυξάνει. Αυτό είναι επόμενο από την στιγμή που έχουμε διπλάσιο αριθμό παραμέτρων. Ο πίνακας μεταβολής του σφάλματος προσεγγίσεως για τέσσερα διαστήματα είναι:

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	-36.79%	-78.24%
3	-85.60%	-98.11%
4	+406.89%	-89.98%
5	+115.69%	-99.79%
6	-0.40%	-93.63%
7	-99.99%	-46.96%
8	-91.86%	-99.71%

Με οκτώ διαστήματα ο πίνακας μεταβολής σφάλματος προσεγγίσεως έχει ως εξής :

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	+2.13%	-94.40%
3	-76.96%	-78.94%
4	+877.17%	-99.96%
5	+18.58%	-70.58%
6	-15.20%	-94.14%
7	-98.91%	-59.57%
8	+22.06%	+80.41%

Για πρώτη φορά στην μέθοδο με προεκπαίδευση CLUSTERING εμφανίζεται η κατάσταση η προσθήκη ενός κόμβου να χειροτερεύει το συνολικό σφάλμα προσεγγίσεως, παρά να το βελτιώνει. Ωστόσο η απόλυτη μεταβολή του λάθους είναι πάρα πολύ μικρή (περίπου 10^{-7}) Αν έχουμε δέκα κόμβους, ο πίνακας μεταβολής του σφάλματος προσεγγίσεως παίρνει την επόμενη μορφή:

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	+2.08%	-97.61%
3	-39.71%	-97.56%
4	-99.38%	-97.52%
5	+22664%	-53.35%
6	-8.94%	-89.78%
7	-99.99%	-86.54%
8	+240.35%	-71.43%

Για δεκαπέντε διαστήματα εκπαιδεύσεως έχουμε τον επόμενο πίνακα μεταβολής σφάλματος προσεγγίσεως:

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	-13.89%	-99.21%
3	+84.62%	-98.15%
4	-81.35%	-98.30%
5	-98.97%	-17.39%
6	-50.00%	-94.74%
7	+34357%	+1600%
8	-99.99%	+252.94%

Τέλος για είκοσι διαστήματα εκπαιδεύσεως η μεταβολή του σφάλματος προσεγγίσεως έχει ως ακολούθως:

ΚΟΜΒΟΙ	TOLMIN	CLUSTERING
1	0.00%	0.00%
2	-96.16%	-99.99%
3	-49.71%	+959.11%
4	+182.29%	-5.59%
5	-75.30%	-99.94%
6	+32.79%	+27.78%
7	-20.99%	-89.13%
8	-53.44%	0.00%

Συμπεράσματα Χρησιμοποιώντας τους πίνακες που είδαμε σε αυτήν την υποενότητα μπορούμε να καταλήξουμε σε κάποια συμπεράσματα:

1. Η προεκπαίδευση με TOLMIN δεν εγγυάται πως με την αύξηση των κόμβων θα έχουμε βελτίωση του σφάλματος προσεγγίσεως.

2. Η προεκπαίδευση με TOLMIN βασίζεται στην αρχικοποίηση των νευρωνικών δικτύων. Το τελικό σφάλμα προσεγγίσεως είναι κάποιο τοπικό ελάχιστο στο οποίο εγκλωβίζονται τα νευρωνικά δίκτυα.

3. Η προεκπαίδευση με CLUSTERING βελτιώνει σημαντικά το σφάλμα προσεγγίσεως καθώς αυξάνουμε το πλήθος των κόμβων.

4. Η αύξηση των διαστημάτων στην προεκπαίδευση με CLUSTERING βελτιώνει και την ταχύτητα πτώσεως του σφάλματος προσεγγίσεως στην γενική περίπτωση.

5. Η προεκπαίδευση με CLUSTERING έχει άσχημα αποτελέσματα μόνον στην περίπτωση που έχουμε πολλά διαστήματα και σημαντικό αριθμό κόμβων. Και σε αυτήν την περίπτωση η αύξηση στο σφάλμα προσεγγίσεως δεν αλλοιώνει την απόλυτη τιμή του σημαντικά, αφού έχουμε να κάνουμε με αυξήσεις της τάξεως του 10^{-7}.

6. Με την χρήση προεκπαιδεύσεως με CLUSTERING εμφανίζεται το φαινόμενο δεκαπέντε διαστήματα να έχουν καλύτερη συμπεριφορά στο σφάλμα προσεγγίσεως από ότι είκοσι και μάλιστα με σχετικά μικρό αριθμό κόμβων. Αυτό σημαίνει πως για να έχουμε ικανοποιητικά αποτελέσματα μπορούμε να χρησιμοποιήσουμε σχετικά μικρό αριθμό παραμέτρων.

7.2 Δύο διαστάσεις

7.2.1 Χαρακτηριστικά πειραμάτων

Στις δύο διαστάσεις σαν συνάρτηση πειραμάτων χρησιμοποιήθηκε η συνάρτηση $\cos x \sin y$ στο διάστημα $[-4, 4] \times [-4, 4]$ της οποίας η γραφική παράσταση δίνεται στο Σχήμα 7.19.

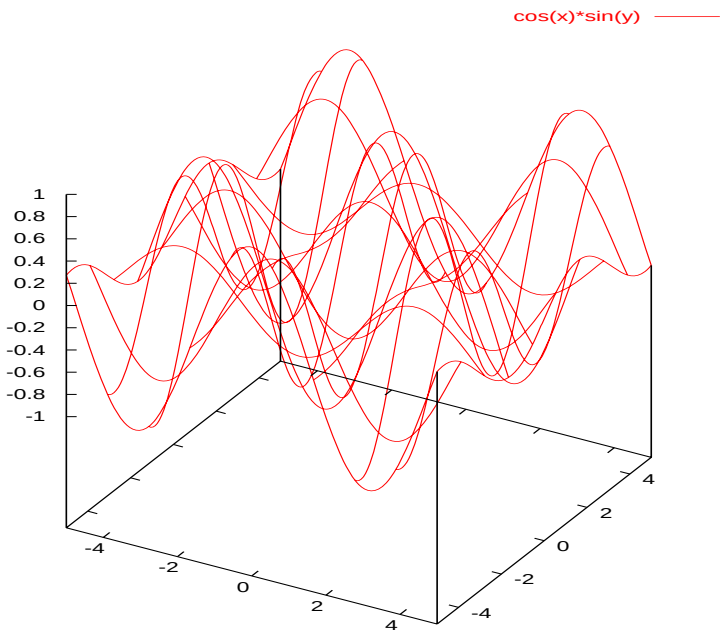
Τα πειράματα που ακολουθούν έγιναν για τις διαμερίσεις 2×2,3×3,, 4×4,, 5×5,, 6×6. Το πλήθος των κόμβων κυμαίνεται από ένα έως οκτώ και χρησιμοποιήθηκε συνέχεια συναρτήσεως. Για προεκπαίδευση χρησιμοποιήθηκε τόσο η TOLMIN όσο και η CLUSTERING.

7.2.2 Προσέγγιση τιμής

7.2.2.1 Προεκπαίδευση με TOLMIN

Το πρώτο θέμα με το οποίο ασχοληθήκαμε στην διδιάστατη περίπτωση ήταν η προσέγγιση τιμής της συναρτήσεως. Στον πίνακα που ακολουθεί εμφανίζονται τα μικρότερα λάθη που πήραμε με την χρήση της μεθόδου TOLMIN για προεκπαίδευση του συστήματος:

ΔΙΑΜΕΡΙΣΗ	ΚΟΜΒΟΙ	ΛΑΘΟΣ
2×2	5	1.2541
3×3	7	0.8835
4×4	8	0.0333
5×5	8	0.2497
6×6	6	0.0237



Σχήμα 7.19: Η γραφική παράσταση της $\cos x \sin y$ στο [-4,4]×[-4,4]

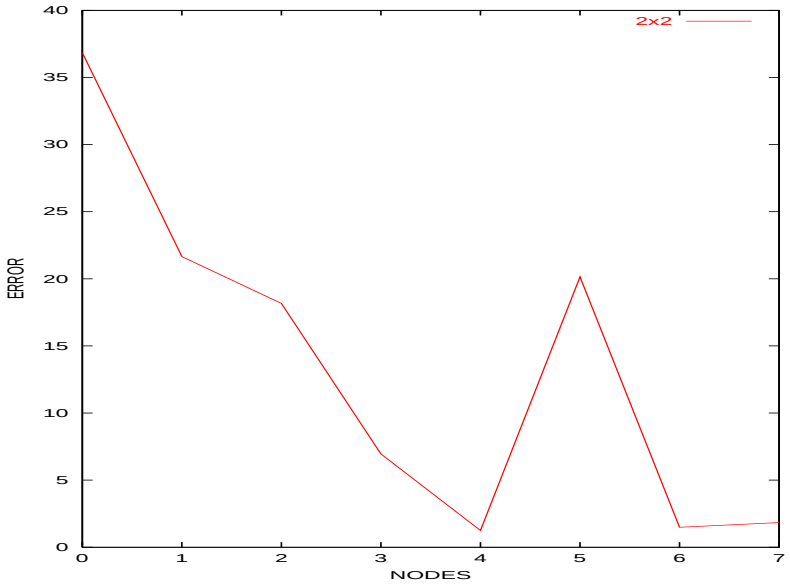
Αν θέλουμε να δούμε γραφικά τα αποτελέσματα για κάθε περίπτωση διαμερίσεως παίρνουμε τα Σχήματα 7.20, 7.21, 7.22, 7.23, 7.24 στα οποία ο οριζόντιος άξονας αναπαριστά τον αριθμό των κρυμμένων κόμβων και ο κατακόρυφος άξονας το τετραγωνικό σφάλμα προσεγγίσεως.

7.2.2.2 Προεκπαίδευση με CLUSTERING

Αν χρησιμοποιήσουμε σαν μέθοδο προεκπαιδεύσεως την μέθοδο CLUSTERING οι μικρότερες τιμές σφάλματος προσεγγίσεως για κάθε κατηγορία διαμερίσεως εμφανίζονται στον επόμενο πίνακα:

ΔΙΑΜΕΡΙΣΗ	ΚΟΜΒΟΙ	Σφάλμα
2×2	8	1.0355
3×3	8	1.0837
4×4	7	0.0242
5×5	7	0.0000001
6×6	8	0.0000001

Οι γραφικές παραστάσεις για κάθε περίπτωση διαμερίσεως εμφανίζονται στα



Σχήμα 7.20: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 2×2 για την συνάρτηση $\cos x \sin y$ με τοπική μέθοδο αρχικοποιήσεως

Σχήματα 7.25, 7.26, 7.27, 7.28, 7.29.

7.2.2.3 Συμπεράσματα

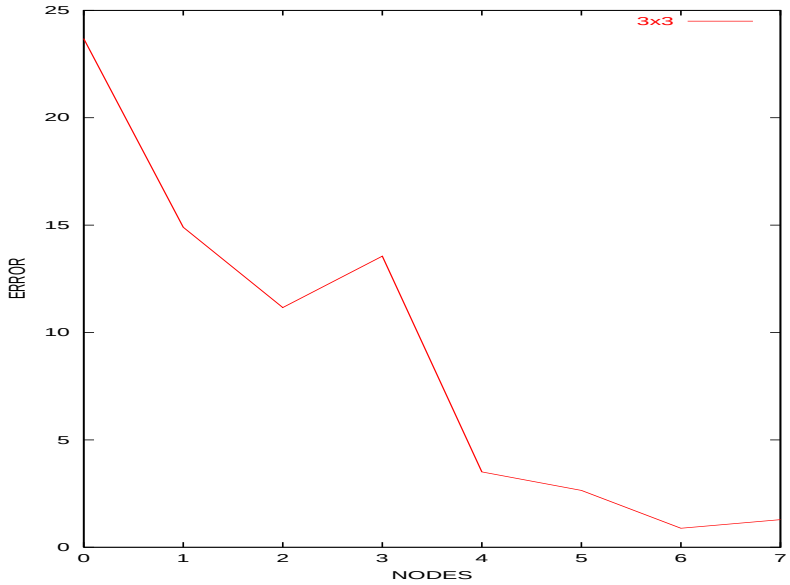
1. Με την μέθοδο TOLMIN είχαμε αρκετά μεγάλες τιμές στο σφάλμα προσεγγίσεως, ακόμα και για πυκνές διαμερίσεις.

2. Στην μέθοδο TOLMIN εμφανίζονται μεγάλες διακυμάνσεις στο σφάλμα προσεγγίσεως, καθώς μεταβάλλεται ο αριθμός των κόμβων. Αυτό το φαινόμενο εμφανίστηκε και στην μία διάσταση και οφείλεται στον εγκλωβισμό των νευρωνικών δικτύων σε τοπικά ελάχιστα της συναρτήσεως σφάλματος.

3. Στην μέθοδο CLUSTERING η πτώση στο σφάλμα προσεγγίσεως είναι περισσότερο ομαλή σε σχέση με την μέθοδο TOLMIN.

4. Η μέθοδος CLUSTERING κατορθώνει να επιλύσει το πρόβλημα ακόμα και με διαμέριση 5×5.

5. Συγκρίνοντας τις καλύτερες περιπτώσεις των δύο αλγορίθμων παίρνουμε τον επόμενο πίνακα, από τον οποίο φαίνεται καθαρά πως η μέθοδος CLUSTERING υπερέχει σχεδόν σε κάθε διαμέριση.



Σχήμα 7.21: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 3×3 για την συνάρτηση $\cos x \sin y$ με τοπική μέθοδο αρχικοποιήσεως

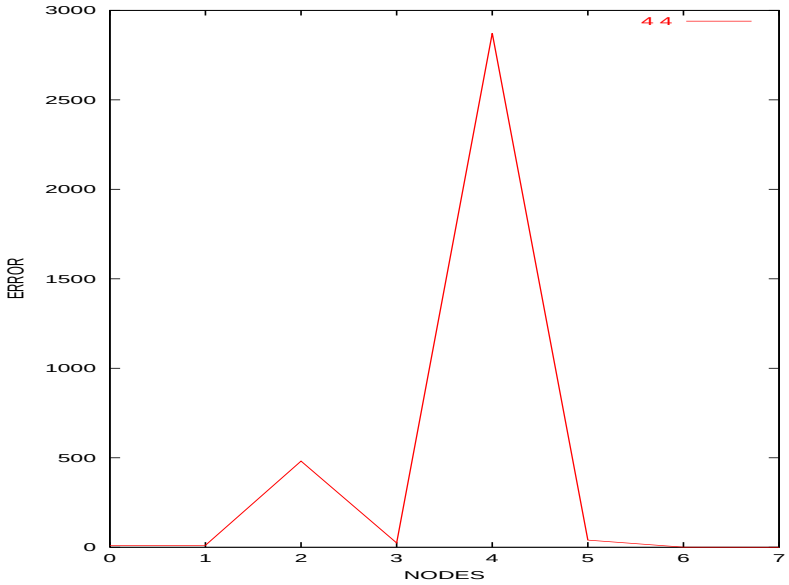
ΔΙΑΜΕΡΙΣΗ	TOLMIN	CLUSTERING	ΔΙΑΦΟΡΑ(%)
2×2	1.2541	1.0355	-17.43%
3×3	0.8835	1.0837	+22.66%
4×4	0.0333	0.0242	-27.33%
5×5	0.2497	0.0000001	-99.99%
6×6	0.0237	0.0000001	-99.99%

7.2.3 Χρόνος εκτελέσεως

7.2.3.1 TOLMIN

Οι χρόνοι εκτελέσεως για τα καλύτερα αποτελέσματα με προεκπαίδευση TOLMIN εμφανίζονται στον επόμενο πίνακα:

ΔΙΑΜΕΡΙΣΗ	ΚΟΜΒΟΙ	ΛΑΘΟΣ	ΧΡΟΝΟΣ
2×2	5	1.2541	288.986
3×3	7	0.8835	601.766
4×4	8	0.0333	1269.302
5×5	8	0.2497	1924.139
6×6	6	0.0237	2771.902



Σχήμα 7.22: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 4×4 για την συνάρτηση $\cos x \sin y$ με τοπική μέθοδο αρχικοποιήσεως

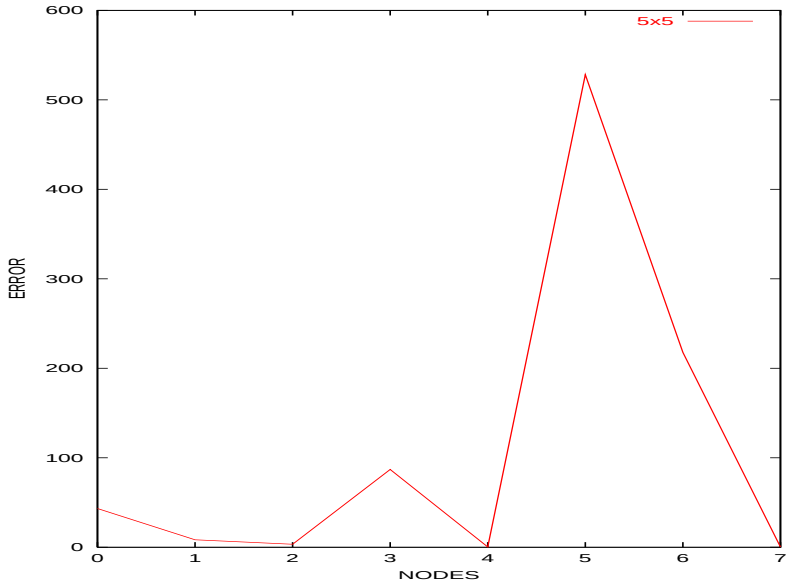
7.2.3.2 CLUSTERING

Ο χρόνος που χρειαστήκαμε με την μέθοδο CLUSTERING για τα μικρότερα σφάλματα προσεγγίσεως εμφανίζεται στον επόμενο πίνακα:

ΔΙΑΜΕΡΙΣΗ	ΚΟΜΒΟΙ	ΛΑΘΟΣ	ΧΡΟΝΟΣ
2×2	8	1.0355	557.026
3×3	8	1.0837	679.143
4×4	7	0.0242	791.346
5×5	7	0.0000001	1023.199
6×6	8	0.0000001	1420.171

7.2.3.3 Συμπεράσματα

Για να μπορέσουμε να συγκρίνουμε τις δύο μεθόδους ως προς τον χρόνο εκτελέσεως κατασκευάζουμε τον επόμενο συγκριτικό πίνακα για τα καλύτερα αποτελέσματα των δύο μεθόδων:



Σχήμα 7.23: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 5×5 για την συνάρτηση $\cos x \sin y$ με τοπική μέθοδο αρχικοποιήσεως

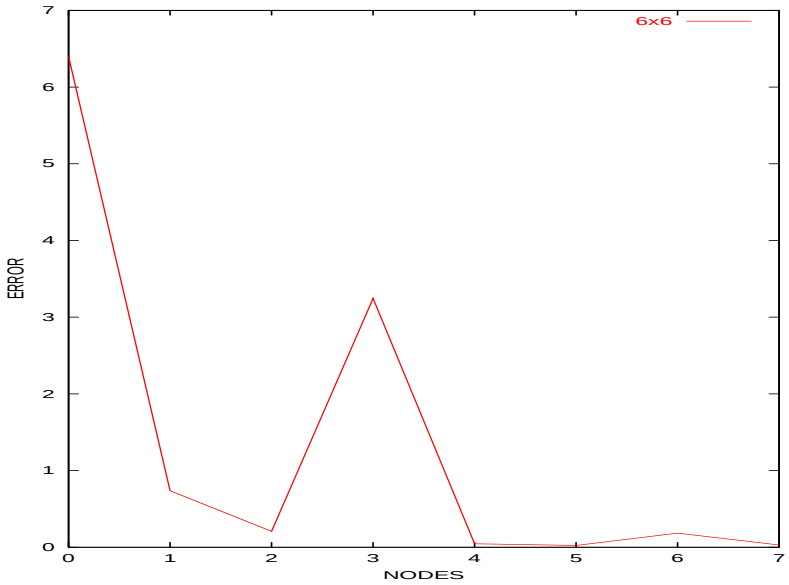
ΔΙΑΜΕΡΙΣΗ	TOLMIN	CLUSTERING	ΔΙΑΦΟΡΑ(%)
2×2	288.986	557.026	+92.75%
3×3	601.766	679.143	+12.86%
4×4	1269.302	791.346	-37.66%
5×5	1924.139	1023.199	-46.82%
6×6	2771.902	1420.171	-48.77%

Από τον παραπάνω πίνακα μπορούμε να εξάγουμε τα ακόλουθα συμπεράσματα:

- Ο χρόνος στην περίπτωση του CLUSTERING μεταβάλλεται πολύ πιο αργά από ότι όταν έχουμε TOLMIN.
- Όταν η μέθοδος με προεκπαίδευση CLUSTERING πλησιάζει ή φτάνει σε λύση, τότε ο χρόνος που χρειαζόμαστε είναι πιο μικρός από ότι όταν χρησιμοποιούμε προεκπαίδευση με TOLMIN.

7.2.4 Πολλοί επεξεργαστές

Όπως και στην περίπτωση της μίας διαστάσεως θα ήταν χρήσιμο να δούμε στις πολλές διαστάσεις την επίπτωση στον χρόνο εκτελέσεως από την χρήση πολλών επεξεργαστών. Τα αποτελέσματα που είχαμε σε αυτήν την περίπτωση ήταν τα ακόλουθα:

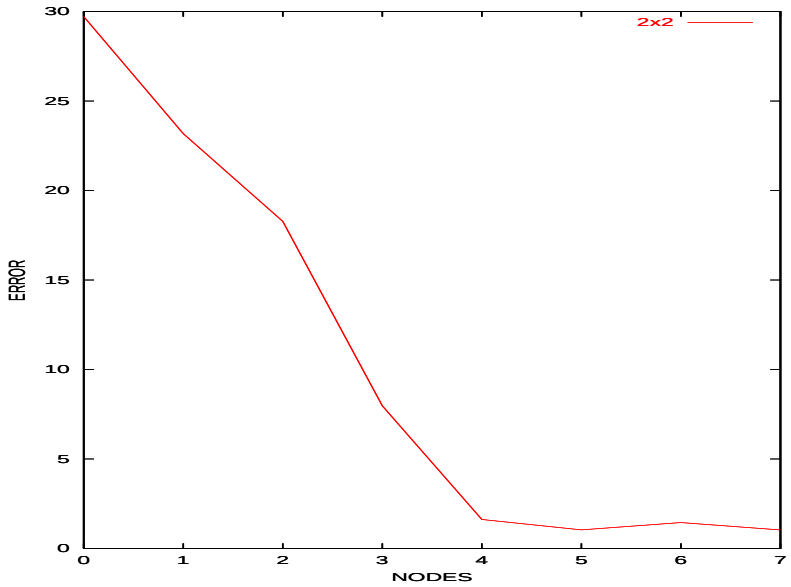


Σχήμα 7.24: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμερίση 6×6 για την συνάρτηση $\cos x \sin y$ με τοπική μέθοδο αρχικοποιήσεως

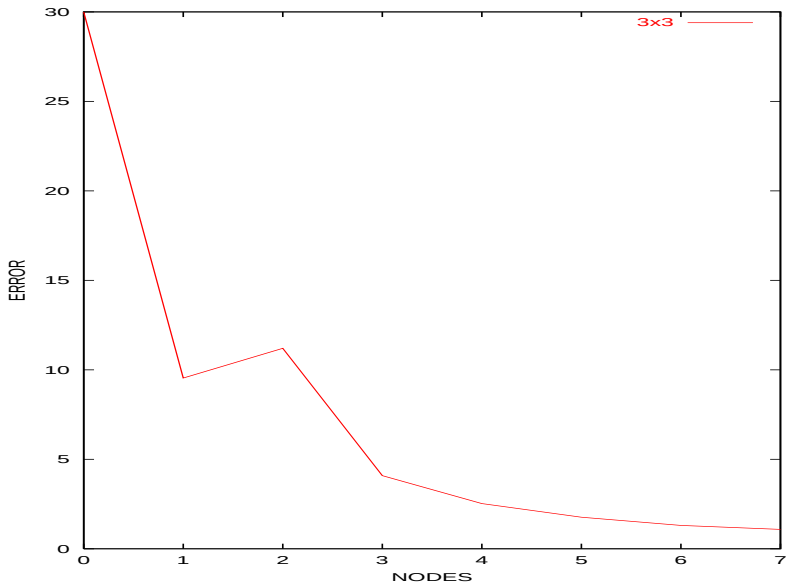
Δ	ΚΟΜΒΟΙ	ΣΦΑΛΜΑ	ΧΡΟΝΟΣ-1	ΧΡΟΝΟΣ-Ν	ΔΙΑΦΟΡΑ(%)
3×3	8	0.52	2563.40	679.14	-73.51%
4×4	7	0.0945	3374.10	791.35	-76.54%
4×4	8	0.0099	2741.06	822.22	-70.00%
5×5	7	0.15	3653.29	1023.20	-71.99%
5×5	8	0.08	3509.77	956.06	-72.76%
6×6	7	0.049	4833.82	1205.40	-75.06%
6×6	8	0.00004	4473.95	1420.17	-68.26%

Η σημασία των στηλών είναι ακριβώς η ίδια με την μονοδιάστατη περίπτωση. Από τον παραπάνω πίνακα μπορούμε να διαπιστώσουμε τα ακόλουθα:

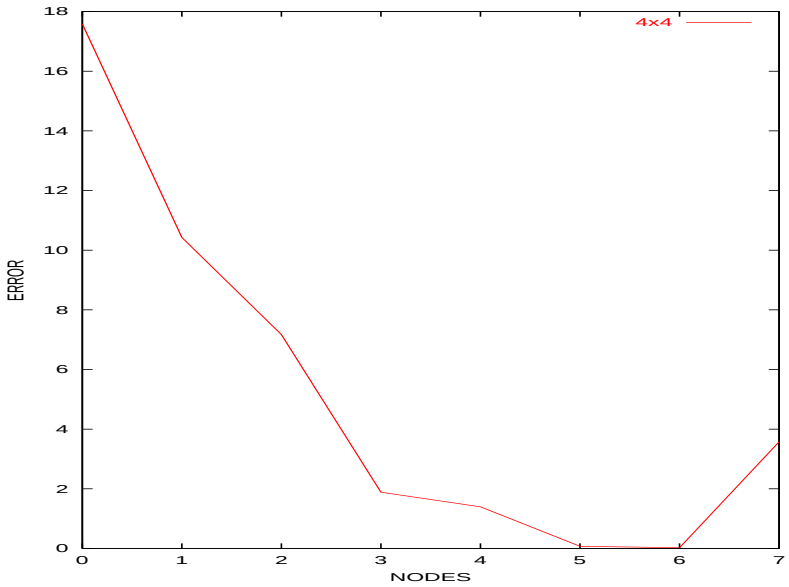
1. Το κέρδος σε ποσοστό χρόνο συνήθως υπερβαίνει το 70%.
2. Με την αύξηση του πλήθους των διαστημάτων δεν υπάρχει ανάλογο κέρδος σε ποσοστό χρόνου. Αυτό συμβαίνει για τους εξής λόγους:
   (α) Το πλήθος των επεξεργαστών που χρησιμοποιήθηκαν στα διδιάστατα πειράματα ήταν περιορισμένο(7).
   (β) Το πλήθος των εξωτερικών μεταβλητών αυξάνει πολύ γρήγορα σε σχέση με τις μεταβλητές κάθε επιμέρους νευρωνικού, καθώς αυξάνεται η διαμέριση. Έτσι υπάρχει ένα μη παράλληλο μέρος στην βελτιστοποίηση που απαιτεί μεγάλο χρόνο για την εκτέλεσή του.



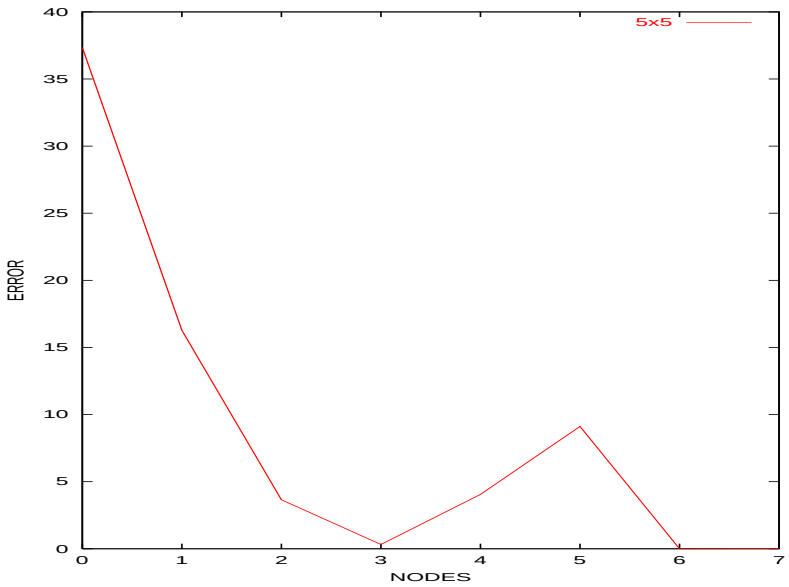
Σχήμα 7.25: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 2×2 για την συνάρτηση $\cos x \sin y$ με καθολική μέθοδο αρχικοποιήσεως



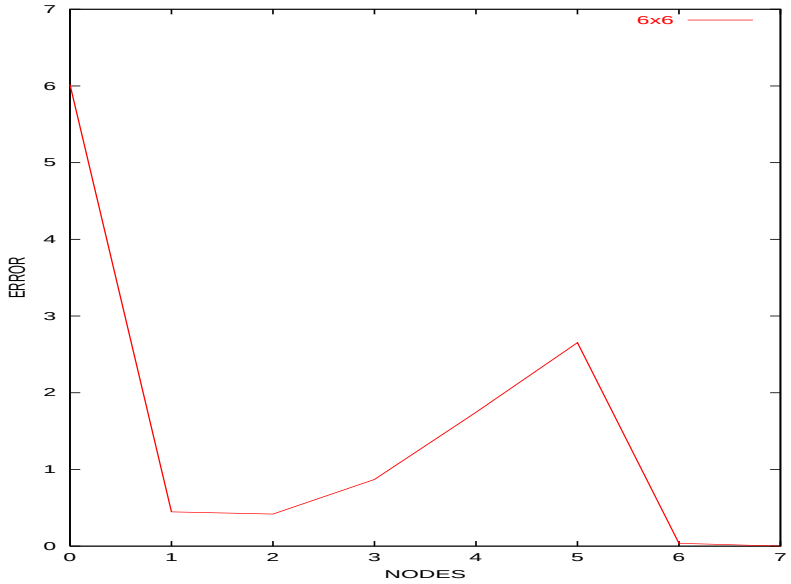
Σχήμα 7.26: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 3×3 για την συνάρτηση $\cos x \sin y$ με καθολική μέθοδο αρχικοποιήσεως



Σχήμα 7.27: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 4×4 για την συνάρτηση $\cos x \sin y$ με καθολική μέθοδο αρχικοποιήσεως



Σχήμα 7.28: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 5×5 για την συνάρτηση $\cos x \sin y$ με καθολική μέθοδο αρχικοποιήσεως



Σχήμα 7.29: Σφάλμα προσεγγίσεως προτεινόμενου μοντέλου στην διαμέριση 6×6 για την συνάρτηση $\cos x \sin y$ με καθολική μέθοδο αρχικοποιήσεως

Κεφάλαιο 8

Μελλοντικές Επεκτάσεις

Στο τελευταίο κεφάλαιο αυτής της αναφοράς θα εξετάσουμε διάφορες επεκτάσεις στο προτεινόμενο μοντέλο.

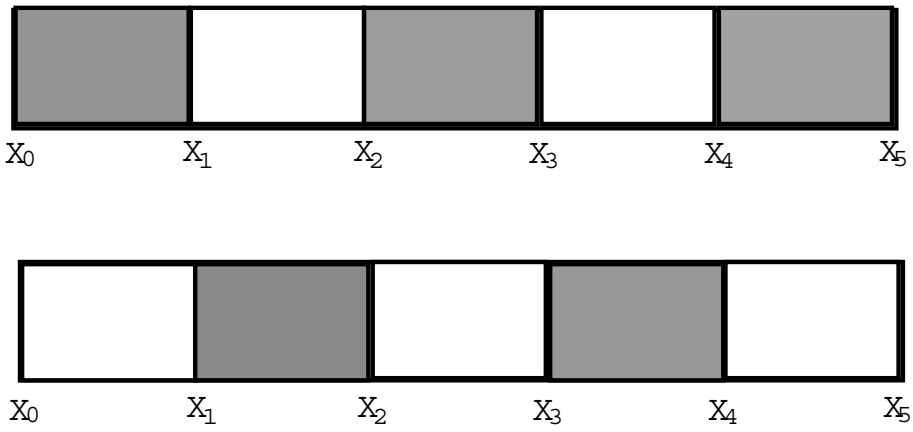
8.1 Αρχικές τιμές

Καθώς διαπιστώσαμε από το προηγούμενο κεφάλαιο υπάρχει μεγάλη σχέση του τελικού αποτελέσματος με τις αρχικές τιμές που επιτυγχάνουμε για τις εξωτερικές παραμέτρους. Μάλιστα όσο καλύτερη είναι αυτή η προσέγγιση τόσο καλύτερα αποτελέσματα επιτυγχάνουμε. Προς το παρόν για να επιτύχουμε καλές αρχικές τιμές εκπαιδεύουμε ένα νευρωνικό δίκτυο στο επιθυμητό σημείο και θεωρούμε την επιστρεφόμενη τιμή ως μία καλή προσέγγιση. Αυτό που διαφοροποιείται στα διάφορα μοντέλα είναι η μέθοδος εκπαιδεύσεως. Σε αυτό που ονομάσαμε τοπική μέθοδος χρησιμοποιήθηκε η παραλλαγή της μεθόδου BFGS με το όνομα TOLMIN. Σε αυτό που ονομάσαμε καθολική μέθοδος χρησιμοποιήθηκε η SINGLE LINKAGE CLUSTERING η οποία περιλαμβάνει πολλές κλήσεις στην BFGS. Από ότι διαπιστώσαμε με την καθολική μέθοδο πετύχαμε πολύ καλά αποτελέσματα χωρίς να χάσουμε σε χρόνο. Ωστόσο θα μπορούσαμε (ίσως) να πετύχουμε καλύτερα αποτελέσματα αν χρησιμοποιούσαμε αρκετά σημεία πέριξ του σημείου ενδιαφέροντος ως σημεία εκπαιδεύσεως για να βρούμε κάποια καλύτερη τιμή για την συνάρτηση στο σημείο ενδιαφέροντος.

8.2 Παράλληλη εκτέλεση

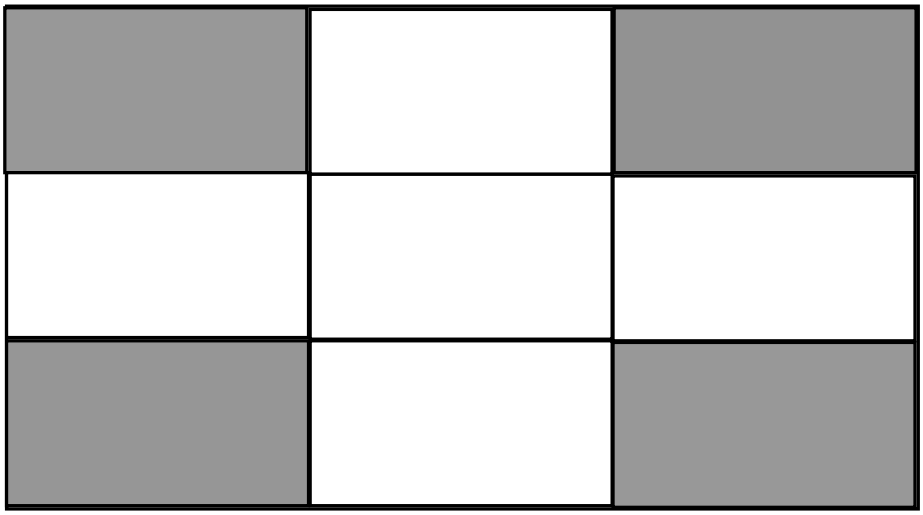
Παρατηρώντας τους πίνακες με το χρονικό κέρδος που είχαμε από την χρήση πολλών επεξεργαστών θα διαπιστώσουμε πως το συνολικό κέρδος δεν υπερβαίνει το 80% παρά το γεγονός πως χρησιμοποιούμε έως και 20 επεξεργαστές. Αυτό συνέβη τόσο εξαιτίας του κόστους επικοινωνίας όσο και του μη παράλληλου μέρους της εργασίας. Για το πρώτο δεν μπορούν να γίνουν και πολλά πράγματα πέρα από το να χρησιμοποιηθεί καλύτερο δίκτυο και υπολογιστικές μονάδες σε

μικρή απόσταση. Για το δεύτερο αίτιο του παραπάνω φαινομένου χρειάζεται μία διαφορετική οργάνωση στον κώδικα. Όπως είναι αυτήν την στιγμή ο κώδικας περιλαμβάνει ένα μη παράλληλο μέρος με το όνομα ΕΞΟΜΑΛΥΝΣΗ, που σαν σκοπό του έχει να βρει καλύτερες τιμές για τις εξωτερικές παραμέτρους βάσει των εκάστοτε τιμών επιμέρους νευρωνικών δικτύων. Αυτό το μέρος περιλαμβάνει μία τοπική εκπαίδευση με πλήθος παραμέτρων που στις δύο διαστάσεις είναι πολλαπλάσιο των παραμέτρων των γειτονικών νευρωνικών δικτύων. Σαν αποτέλεσμα υπάρχει μία σταθερή επιβάρυνση στον χρόνο. Αυτό που θα μπορούσε να προταθεί εδώ είναι η **εκπαίδευση των μονών - ζυγών**. Στην μία διάσταση αυτό μπορεί να αποδοθεί με το Σχήμα 8.1:



Σχήμα 8.1: Μονά - Ζυγά μίας διαστάσεως

Στο παραπάνω σχήμα έχουμε πέντε διαστήματα που έχουν χαρακτηριστεί με δύο χρώματα: γκρίζο και λευκό. Αυτά που έχουν γκρίζο χρώμα εκπαιδεύονται μαζί ως προς τις αντίστοιχες εξωτερικές παραμέτρους στα σημεία τομής των διαστημάτων και αυτά που έχουν λευκό χρώμα θα εκπαιδευτούν στην επόμενη φάση, όπου και αλλάζουν τα χρώματα. Με αυτόν τον τρόπο επιτυγχάνεται ο μέγιστος παραλληλισμός, αφού δεν υπάρχουν κοινές μεταβλητές ανάμεσα στα εκπαιδευόμενα μοντέλα. Στις δύο διαστάσεις έχουμε το επόμενο σχήμα μονών ζυγών:



Σχήμα 8.2: Μονά - Ζυγά δύο διαστάσεων

Φυσικά το παραπάνω σχήμα δεν είναι μοναδικό, όπως δεν είναι και μοναδική η επόμενη κατάσταση στην οποία μπορούμε να μεταβούμε μετά από αυτό.

8.3 Διαφορικές εξισώσεις

Μία άλλη επέκταση που μπορεί κάποιος να σκεφτεί για το σύστημα που μελετήσαμε σε αυτήν την εργασία είναι η χρήση του για την επίλυση μερικών διαφορικών εξισώσεων.

Βιβλιογραφία

[1] Gisela Engeln-Mullges and Frank Uhlig. "Numerical Algorithms with Fortran". Springer Verlang Publications, Berlin, 1996.

[2] William H. Press Brian P. Flannery and Saul A. Teukolsy. "Numerical Recipes the Art of Scientific Computing". Cambridge University Press, Cambridge, 1986.

[3] D. G. Papageorgiou I. N. Demetropoulos and I. E. Lagaris. "Merlin 3.0, A Multidimensional Optimization Environment". Comput. Phys. Commun. 109, p.p. 227-249, 1998.

[4] A. H. Rinnooy Kan and G. T. Timmer. "Stochastic global optimization methods methods-Part I: clustering methods". Report 85391A, Econometric Institute, Erasmus University, The Netherlands, 1985.

[5] Γ. Δ. Ακρίβης και Β. Α. Δουγαλής. "Εισαγωγή στην Αριθμητική Ανάλυση". Πανεπιστημιακές Εκδόσεις Κρήτης, Ηράκλειο, 1997.

[6] Ισαάκ Η. Λαγαρής. "Αριθμητικές Μέθοδοι Βελτιστοποίησης Θεωρία και Λογισμικό". Πανεπιστήμιο Ιωαννίνων, Ιωάννινα, 1999.

[7] D. G. Papageorgiou, I. N. Demetropoulos and I. E. Lagaris. "The Merlin Control Language for strategic optimization". Comput. Phys. Commun. 109, p.p. 250-270, 1998.

[8] Christopher M. Bishop. "Neural Networks for Pattern Recognition". Clarendon Press, Oxford, 1995.

[9] Hank Dietz. "Linux Parallel Processing HOWTO". Purdue University, 1998

[10] Peter S. Pacheco. "A User's Guide to MPI". University of San Francisco, San Fransisco, 1998.

[11] G. Cybenko. "Approximation by superpositions of a sigmoidal function". Mathematics of Control Signals and Systems 2, 1969.

[12] N. Obreshkov. "On the Mechanical Quadratures". J. Bulgar. Acad. Sci. and Arts LXV-8, 1942

Σχήματα