ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΓΕΝΙΚΟ ΤΜΗΜΑ

ΕΠΙΣΤΗΜΟΝΙΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ ΣΕ ΠΑΡΑΛΛΗΛΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

Εμμανουήλ Μαθιουδάκης

Εργαστήριο Εφαρμοσμένων Μαθηματικών
και Ηλεκτρονικών Υπολογιστών

Διπλωματική Διατριβή
Μεταπτυχιακού Διπλώματος Ειδίκευσης

Επιβλέπων Καθηγητής : Αν. Καθηγ. Ι. Σαριδάκης

Χανιά Ιανουάριος 1996

Ευχαριστίες

Πρώτα θα ήθελα να ευχαριστήσω το σύμβουλο καθηγητή μου, Αναπληρωτή Καθηγητή του Γενικού Τμήματος Γιάννη Γ. Σαριδάκη, ο οποίος μου παρείχε την άρτια επιστημονική καθοδήγηση και την ηθική υποστήριξη στην ολοκλήρωση της παρούσας Διατριβής.

Την Επίκουρο Καθηγήτρια Ελενα Παπαδοπούλου ευχαριστώ για την πολύτιμη συνεισφορά της στην επιστημονική καθοδήγηση.

Η εκπόνηση αυτής της εργασίας θα ήταν αδύνατη χωρίς τον τεχνολογικό εξοπλισμό του Εργαστηρίου Εφαρμοσμένων Μαθηματικών και Ηλεκτρονικών Υπολογιστών και γι' αυτό θα ήθελα να ευχαριστήσω τον διευθυντή του Εργαστηρίου Καθηγητή Γεώργιο Αβδελά.

Για ένα σημαντικό μέρος της Διατριβής ήταν απαραίτητη η χρήση παράλληλου υπολογιστικού συστήματος. Ευχαριστώ τον Καθηγητή Σταύρο Χριστοδουλάκη για την παραχώρηση πρόσβασης στο παράλληλο σύστημα του Εργαστηρίου Διανεμημένων Συστημάτων και Εφαρμογών.

Επίσης θα ήθελα να ευχαριστήσω τον διδάκτορα Βασίλη Παπαγεωργίου για την παροχή πολύτιμων επιστημονικών γνώσεων.

Οι συνάδελφοι μου Μαρία Ζακυνθινάκη και Βασιλική Κακαβά, που μαζί περάσαμε δύσκολες ώρες κατά τη διάρκεια των μεταπτυχιακών μας σπουδών, αξίζουν ιδιαίτερης μνείας.

Τέλος ευχαριστώ τα μέλη της οικογενείας μου για την ηθική υποστήριξη που μου παρείχαν.

Περίληψη

Τα τελευταία χρόνια ο **παραλληλισμός** (parallelism) αναδείχθηκε σ' έναν από τους σημαντικότερους παράγοντες για την ανάπτυξη της **αποδοτικότητας**. Γι' αυτό και ο υπολογισμός μεγάλης - κλίμακας αριθμητικών προβλημάτων σε παράλληλα περιβάλλοντα έχει γίνει ένα από τα κεντρικά θέματα της Αριθμητικής Ανάλυσης και των Επιστημονικών Υπολογισμών (Scientific Computing) γενικότερα. Επειδή όμως οι κλασσικοί αριθμητικοί αλγόριθμοι αδυνατούν να εκμεταλλευτούν τις δυνατότητες των παράλληλων μηχανών, κύρια έμφαση στην έρευνα παράλληλων αλγορίθμων για μεγάλης κλίμακας προβλημάτων, έχει δοθεί στην ανάπτυξη τεχνικών παραλληλοποίησης των σειριακών αλγορίθμων, έτσι ώστε η απεικόνισή τους σε παράλληλες αρχιτεκτονικές να είναι βέλτιστα αποδοτική.

Αριθμητικές μέθοδοι για την επίλυση Διαφορικών Εξισώσεων με Μερικές Παραγώγους πάντοτε βρίσκονταν στο θεματολογικό επίκεντρο των Επιστημονικών Υπολογισμών, προσελκύοντας ένα πολύ μεγάλο τμήμα του ερευνητικού ενδιαφέροντος. Στην κατηγορία αυτή ανήκει και η μέθοδος Πεπερασμένων Στοιχείων Collocation, η οποία έχει αποδειχθεί από τις πλέον αποδοτικές για την επίλυση, κυρίως Ελλειπτικών, Προβλημάτων Συνοριακών Τιμών (ΠΣΤ). Και επειδή το μέγεθος του αντίστοιχου γραμμικού συστήματος, το οποίο προκύπτει από τη διακριτοποίηση του ΠΣΤ μέσω της collocation, είναι πολύ μεγάλης τάξης, η χρήση επαναληπτικών μεθόδων για την επίλυσή του θεωρείται επιβεβλημένη.

Στο πνεύμα όλων των παραπάνω ο κύριος στόχος της παρούσας εργασίας είναι :

- Η παραλληλοποίηση της επαναληπτικής μεθόδου AOR, όταν αυτή εφαρμόζεται για την επίλυση μεγάλης τάξης γραμμικών συστημάτων, τα οποία προκύπτουν από τη διακριτοποίηση του Poisson - Dirichlet προβλήματος μέσω της μεθόδου Collocation με Hermite Bicubic στοιχεία.
- Η απεικόνιση του προκύπτοντος παράλληλου αλγορίθμου σε
  1. Λογικού τύπου Παράλληλες Μηχανές [Virtual Parallel Machines (PVM)].
  2. Διανεμημένης Μνήμης (Distributed Memory) MIMD Παράλληλες Μηχανές.

Για την αντιμετώπιση του προβλήματος της παραλληλοποίησης του σειριακού αλγορίθμου χρησιμοποιήσαμε με επιτυχία τις ακόλουθες τεχνικές

1. **Επαναδιαμέριση** του collocation πίνακα χρησιμοποιώντας μετασχηματισμούς ομοιότητας
2. **Preconditioning** του collocation πίνακα

για να αυξήσουμε τον βαθμό του παραλληλισμού σε βέλτιστο σημείο. Ο προκύπτων παράλληλος αλγόριθμος εφαρμόστηκε σε μια PVM παράλληλη μηχανή αποτελούμενη από τρεις servers RISC αρχιτεκτονικής του Εργαστηρίου Εφαρμοσμένων Μαθηματικών και Ηλεκτρονικών Υπολογιστών, καθώς και στον Transputer based παράλληλο υπολογιστή του Εργαστηρίου Διανεμημένων Συστημάτων και Εφαρμογών.

Συμπερασματικά σημειώνουμε ότι, ενώ στην PVM εφαρμογή το κόστος του αργού δικτύου μειώνει την απόδοση του αλγορίθμου, όπως άλλωστε αναμενόταν, στην περίπτωση της MIMD παράλληλης αρχιτεκτονικής ασυμπτωτικά τείνουμε στο θεωρητικό βέλτιστο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Σελ.

1 PARALLEL VIRTUAL MACHINE

1.1 Εισαγωγή

Το PVM (Parallel Virtual Machine) είναι ένα πακέτο λογισμικού, το οποίο παρέχει την δυνατότητα σε μια συλλογή από υπολογιστικά συστημάτα, διαφορετικών αρχιτεκτονικών, να λειτουργήσουν σαν μια ολοκληρωμένη παράλληλη μηχανή.

Τα επιμέρους υπολογιστικά συστήματα μπορεί να απαρτίζονται από shared ή distributed memory υπολογιστές, vector supercomputers, scalar workstations ή εξιδικευμένα μηχανήματα γραφικών, ενώ επιτρέπεται να είναι συνδεμένα μεταξύ τους σε δίκτυα με ένα ευρύ φάσμα διαφορετικών συνδέσεων, όπως με καλώδια τύπου Ethernet, οπτικών ινών, κτλ.
Το λογισμικό του PVM υποστηρίζει σε κάθε μηχάνημα ένα ξεχωριστό περιβάλλον εργασίας, το οποίο ο κάθε χρήστης μπορεί να το προσαρμόσει ανάλογα με τις απαιτήσεις της αναπτυσσόμενης εφαρμογής.

Υποστηρίζονται προγράμματα γραμμένα σε C ή Fortran και η σύνδεση τους με το PVM επιτυγχάνεται με τη χρήση διαδικασιών και συναρτήσεων, οι οποίες πε-

ριλαμβάνονται στην ειδική βιβλιοθήκη υποπρογραμμάτων libpvm και επιτρέπουν την ενεργοποίηση εφαρμογών, ανταλλαγή μηνυμάτων και δεδομένων και γενικότερα συγχρονισμού της λειτουργίας της παράλληλης υπολογιστικής μηχανής.

Μέσω του PVM οι χρήστες έχουν το δικαίωμα της επιλεκτικής εκτέλεσης τμημάτων των εφαρμογών στα περισσότερο κατάλληλα επιμέρους υπολογιστικά συστήματα.

Για το PVM προτιμούνται εφαρμογές, οι οποίες διαχωρίζονται σε ανεξάρτητα μεταξύ τους τμήματα ή και σε σειριακού τύπου αλγορίθμους, στους οποίους κάποιο τμήμα "ταιριάζει" περισσότερο σε κάποιο από τα υπολογιστικά συστήματα της δημιουργούμενης παράλληλης μηχανής.

Το PVM έχει δοκιμαστεί με ανάπτυξη εφαρμογών, οι οποίες ανήκουν σε ευρύ φάσμα ερευνητικών περιοχών, όπως στη τεχνολογία υλικών, μοντελοποίηση κλιματολογικών συνθηκών, μετεωρολογία, εξομοίωση κινήσεων σωματιδίων, δι-διάστατα και τριδιάστατα γραφικά, σεισμολογία, μελέτη διαστημικών μοντέλων και υδροακουστική [6].

Παράλληλα, με το PVM έχουν αναπτυχθεί και συνεχίζουν να εξελίσσονται μαζί του, ορισμένες X-window εφαρμογές, οι οποίες το συμπληρώνουν ανάλογα με τις απαιτήσεις των χρηστών. Τα βασικότερα από τα συμπληρωματικά αυτά εργαλεία [1, 2, 8] είναι το HeNCE, το οποίο είναι ένα Graphical User Interface (GUI) για προγραμματισμό και επίβλεψη της εκτέλεσης των εφαρμογών σε δίκτυο υπολογιστών μέσω του PVM και το XPVM [1] με το οποίο ο χρήστης έχει τη δυνατότητα του ελέγχου της παράλληλης μηχανής, μαζί με την επίβλεψη και τη μελέτη της εκτέλεσης των εφαρμογών.

Αν και το PVM εξακολουθεί να βρίσκεται σε ερευνητική μορφή από το 1989, όπου ξεκίνησε η ανάπτυξή του, έχει εξαπλωθεί σ' όλο το κόσμο με χιλιάδες ερευνητές να το χρησιμοποιούν συστηματικά. Τα αποτελέσματα των ερευνών τους καθώς επίσης και οι νέες προοπτικές σχετικές με το PVM παρουσιάζονται σε ετήσια διεθνή συνέδρια.

1.2 PVM - Parallel Virtual Machine

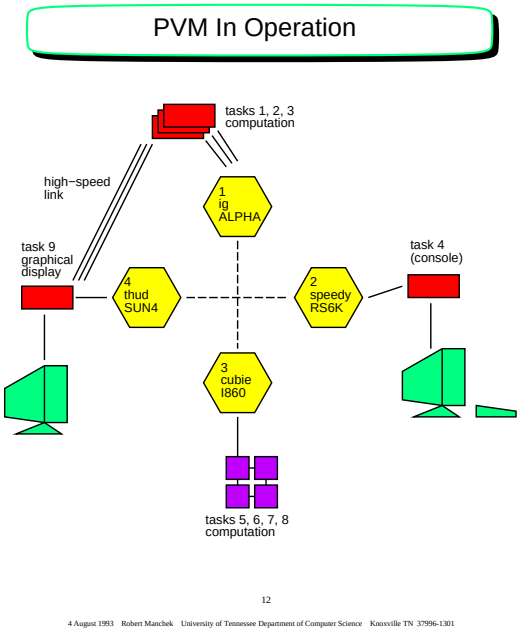
Το λογισμικό πακέτο PVM 3 μετατρέπει ένα δίκτυο UNIX υπολογιστικών συστημάτων με διαφορετικές αρχιτεκτονικές σε μια ολοκληρωμένη παράλληλη μηχανή.
Σε τέτοιου είδους παράλληλους ηλεκτρονικούς υπολογιστές μπορούν να αναπτυχθούν πολύπλοκες εφαρμογές αξιοποιώντας την συκεντρωμένη υπολογιστική ισχύ, η οποία ήταν κατανεμημένη στα επιμέρους υπολογιστικά συστήματα.

Η ανάπτυξη του PVM ξεκίνησε το καλοκαίρι του 1989 στο Oak Ridge National Laboratory (ORNL) και εξελίσσεται μέχρι σήμερα με τη συνεργασία ερευνητών και από τα Emory University, University of Tennessee (UT), Carnegie Mellon University και Pittsburgh Supercomputer Center.

Το PVM δημιουργήθηκε από την ανάγκη της αξιοποίησης της υπολογιστικής ισχύς, που βρίσκεται κατανεμημένη σε κάθε είδους μηχάνημα σ' ένα υπολογιστικό κέντρο.
Τα μηχανήματα που έχουν δοκιμαστεί για αυτό το σκοπό μπορεί να είναι :

- Workstations και PCs με λειτουργικό σύστημα τύπου Unix
- Distributed - memory multiprocessors όπως Intel hypercubes
- Shared - memory multiprocessors όπως SGI Challenge

Η πρώτη κατηγορία μηχανημάτων χρησιμοποιεί το πρωτόκολλο επικοινωνίας TCP/IP για μεταφορά δεδομένων, η δεύτερη native message - passing υπορουτίνες για μεταφορά δεδομένων στους κόμβους και η τρίτη τη κοινή μνήμη.
Επιτρέπεται και η συμμετοχή μηχανημάτων με διαφορετικο από το Unix λειτουργικό σύστημα, όπως το VMS της DEC και το OS/2 της IBM. Ένα παράδειγμα μιας PVM μηχανής δίνεται στην Εικόνα 1.
Η δημιουργούμενη παράλληλη μηχανή είναι λογικού τύπου κι έτσι χαρακτηρίζεται με τον όρο **virtual machine**. Τα επιμέρους υπολογιστικά συστήματα



Εικόνα 1 : PVM σε λειτουργία

αναφέρονται σαν **hosts**, ενώ συχνά με τον όρο **task** εννοείται μια PVM μονάδα υπολογισμού.

Προγράμματα γραμμένα σε Fortran 77 ή σε C μπορούν να εκτελεστούν παράλληλα, αφού μέσω διαδικασιών και συναρτήσεων του PVM επιτυγχάνεται η μετάδοση και λήψη μηνυμάτων, ώστε να συνεργαστούν πολλαπλά tasks μεταξύ τους και να επιλυθεί κάποιο πρόβλημα.

Στο PVM μπορούν να συμμετέχουν μηχανήματα με διαφορετικές παραστάσεις αριθμών μηχανής κι έτσι είναι εφικτή η εκτέλεση κάθε task στη κατάλληλη αρχιτεκτονική.

Το PVM απαρτίζεται από δύο κύρια τμήματα. Το πρώτο είναι το πρόγραμμα $pvmd3$, το οποίο συχνά αναφέρεται και ως **pvmd** και τοποθετείται σε κάθε host της Parallel Virtual Machine. Το pvmd έχει σχεδιαστεί, ώστε να μπορεί να εγκατασταθεί από ένα κοινό χρήστη σε κάθε μηχάνημα. Όταν κάποιος χρήστης θελήσει να εκτελέσει μια PVM εφαρμογή, αρχικά δημιουργεί την δική του Parallel

Virtual Machine θέτοντας σε λειτουργία το pvmd3 σε κάθε μηχάνημα και στη συνέχεια εκτελεί την εφαρμογή από το unix prompt οποιουδήποτε host.
Πολλαπλοί χρήστες έχουν την δυνατότητα να συνθέσουν αλληλοκαλυπτόμενες Parallel Virtual Machines κι έτσι κάθε χρήστης είναι σε θέση να εκτελεί ταυτόχρονα διαφορετικές εφαρμογές στο δικό του PVM σύστημα.

Το δεύτερο τμήμα του PVM είναι η βιβλιοθήκη **libpvm3.a**, η οποία περιέχει υποπρογράμματα, τα οποία είναι σχετικά με την μεταφορά δεδομένων ανάμεσα στους hosts, έναρξη διαδικασιών και γενικά συντονισμού της λειτουργίας της virtual machine. Όλες οι εφαρμογές του PVM είναι απαραίτητο, στο στάδιο της μεταγλώττισης, να συνδέονται με την βιβλιοθήκη αυτή.

Υπάρχει η δυνατότητα της επικοινωνίας με τους δημιουργούς του PVM για κάθε είδους πρόβλημα μέσω του ηλεκτρονικού ταχυδρομείου στην διεύθυνση **pvm@msr.epm.ornl.gov**. Οι χρήστες του PVM μπορούν να ενημερώνονται για τις τελευταίες εξελίξεις στο USENET newsgroup **comp.parallel.pvm**.

1.3 Απόκτηση και Εγκατάσταση του PVM

Η εγκατάσταση του PVM δεν εμφανίζει δυσκολίες και δεν απαιτεί να διαθέτει ο χρήστης αυξημένες αρμοδιότητες στο σύστημα [9]. Ένας απλός χρήστης μπορεί να το εγκαταστήσει σε κάθε host και για τους υπόλοιπους. Αυτό αποτελεί έναν από τους βασικότερους λόγους της μεγάλης εξάπλωσης του PVM.

α. Απόκτηση του Λογισμικού

Ολόκληρο το πακέτο του PVM μαζί με τα κατάλληλα εγχειρίδια προσφέρεται ελεύθερα σ'οποιονδήποτε χρήστη του Internet από τον server *netlib* μέσω ftp, WWW, xnetlib ή e-mail.

Οι τρόποι απόκτησης του Λογισμικού μπορεί να είναι οι ακόλουθοι :

- Με anonymous ftp στη διεύθυνση **netlib2.cs.utk.edu** και συγκεκριμένα στο directory pvm3 υπάρχει το αρχείο index, το οποίο αναφέρει όλα τα σχετικά αρχεία που περιέχονται.

- Μέσω της χρήσης του αντίστοιχου World Wide Web server με διεύθυνση **http://www.netlib.org/pvm3/index.html**.

- Κάνοντας χρήση της X-Window εφαρμογής Xnetlib, η οποία επιτρέπει την σύνδεση, ενημέρωση και μεταφορά αρχείων από την netlib.
  Τα σχετικά αρχεία της Xnetlib υπάρχουν στο directory pub/xnetlib.

- Στέλνοντας e-mail στη διεύθυνση netlib@ornl.gov με περιεχόμενο το παρακάτω μήνυμα : **send index from pvm3**. Αυτόματα ο χρήστης θα λάβει ένα αρχείο με τις λίστες των περιεχόμενων αρχείων και τις κατάλληλες οδηγίες για την απόκτησή τους, μέσω του ηλεκτρονικού ταχυδρομείου.

Το λογισμικό λαμβάνεται σε συμπιεσμένη μορφή με διάφορα είδη συμπίεσης. Συνήθως η αποσυμπίεση επιτυγχάνεται με τη χρήση του προγράμματος gunzip, στη περίπτωση κατά την οποία το αποσυμπιεζόμενο αρχείο έχει κατάληξη *.gz. Μετά την αποσυμπίεση προκύπτει ένα tar αρχείο, το οποίο θα πρέπει να γίνει untar, αφού μεταφερθεί στη θέση που έχει επιλεγεί στο σύστημα για την δημιουργία του pvm3 directory. Αυτό επιτυγχάνεται με την εντολή
tar xvf *.tar.

β. Εγκατάσταση

Για τους πιο διαδεδομένους τύπους μηχανημάτων, υπάρχουν διαθέσιμα τα κατάλληλα εκτελέσιμα αρχεία. Αν ο τύπος του μηχανήματος δεν είναι ανάμεσα σ' αυτά, θα πρέπει να δημιουργηθούν τα εκτελέσιμα μαζί με τις βιβλιοθήκες από τα C source αρχεία, τα οποία μπορούν να αποκτηθούν με παρόμοια διαδικασία.

Πριν την εγκατάσταση του PVM είναι απαραίτητο να ορισθούν δυο βασικές μεταβλητές στο περιβάλλον εργασίας του χρήστη. Οι μεταβλητές αυτές ορίζονται στα κατάλληλα αρχεία ανάλογα με το είδος του shell, όπως είναι το .login ή .profile.

Οι βασικότερες μεταβλητές που πρέπει να ορίσει κάθε χρήστης στο περιβάλλον εργασίας του σε κάθε host είναι η PVM_ARCH, η οποία καθορίζει το είδος της αρχιτεκτονικής κάθε επιμέρους υπολογιστικού συστήματος και PVM_ROOT η οποία καθορίζει το σημείο που βρίσκεται το directory pvm3 στο σύστημα.

Στον πίνακα 1 αναφέρονται όλα τα είδη των αρχιτεκτονικών που υποστηρίζονται από το PVM 3 καθώς και η τιμή της μεταβλητής PVM_ARCH για κάθε αρχιτεκτονική.

Μετά την επιτυχή εγκατάσταση του PVM σε κάθε υποψήφιο μέλος της παράλληλης μηχανής, χρειάζεται να δημιουργηθεί στο home directory κάθε χρήστη ένα directory, συνήθως το pvm3/bin/$PVM_ARCH/, στο οποίο θα τοποθετούνται τα εκτελέσιμα αρχεία. Αν η ονομασία του directory είναι διαφορετική θα πρέπει να χρησιμοποιηθεί η μέθοδος των hostfiles, ώστε να ενημερωθεί κατάλληλα το pvm.

Επίσης είναι απαραίτητο να υπάρχει στο $HOME directory κάθε χρήστη του pvm ένα αρχείο με την ονομασία .rhosts, το οποίο θα αναφέρει τα ονόματα των hosts που θα συμμετέχουν στη παράλληλη μηχανή.

Η παραπάνω μέθοδος εφαρμόζεται στην περίπτωση κατά την οποία ο χρήστης διαθέτει πρόσβαση σε κάθε host με το ίδιο username. Αν το username είναι διαφορετικό θα πρέπει να χρησιμοποιηθεί η μέθοδος των hostfiles, ώστε να ενημερωθεί το pvm και να ζητήσει το password του χρήστη.

Πιθανά προβλήματα τα οποία μπορεί να προκύψουν, όπως η μη ενεργοποίηση του pvmd σε κάποιο host, αντιμετωπίζονται με τη χρήση των hostfiles. Τα αρχεία

PVM_ARCH	Machine	Notes
AFX8	Alliant FX/8	
ALPHA	DEC Alpha	DEC OSF-1
BAL	Sequent Balance	DYNIX
BFLY	BBN Butterfly TC2000	
BSD386	80386/486 Unix box	BSDI
CM2	Thinking Machines CM2	Sun front-end
CM5	Thinking Machines CM5	
CNVX	Convex C-series	
CNVXN	Convex C-series	native mode
CRAY	C-90,YMP	UNICOS
CRAY2	Cray-2	
CRAYSMP	Cray S-MP	
DGAV	Data General Aviion	
E88K	Encore 88000	
HP300	HP-9000 model 300	HPUX
HPPA	HP-9000 PA-RISC	
I860	Intel iPSC/860	link-lrpc
IPSC2	Intel iPSC/2 386 host	SysV
KSR1	Kendall Square KSR-1	OSF-1
LINUX	80486 Pentium - LINUX box	LINUX
MASPAR	MASPAR host	
MIPS	MIPS 4680	
NEXT	NeXT	
PGON	Intel Paragon	link -lrpc
PMAX	DECstation 3100,5100	Ultrix
RS6K	IBM/RS6000	AIX
RT	IBM RT	
SGI	Silicon Graphics	IRIX 4.x
SGI5	Silicon Graphics	IRIX 5.1
SUN3	Sun 3	SunOS 4.2
SUN4	Sun 4,SPARCstation	SunOS 4.2
SUN4SOL2	Sun 4,SPARCstation	Solaris 2.2
SYMM	Sequent Symmetry	
U370	IBM 370	AIX
UVAX	DEC MicroVAX	

Πίνακας 1 : Τιμές της μεταβλητής PVM_ARCH.

αυτά χρησιμοποιούνται για να ορίσουν κάποιες παραμέτρους του pvm σύμφωνα με τις ιδιαιτερότητες της δημιουργούμενης παράλληλης μηχανής.

Μετά την ολοκλήρωση του ορισμού των μεταβλητών του pvm θα πρέπει να επιλεγεί ένα από τα host μηχανήματα, το οποίο θα ρυθμίζει την λειτουργία του συστήματος.
Πρόκειται για ένα stand alone PVM task, το οποίο ονομάζεται pvm console και ενεργοποιείται με την εντολή pvm $[hostfile]$, μέσω της οποίας ο χρήστης ελέγχει τη παράλληλη μηχανή. Το task αυτό μπορεί να ξεκινήσει ή να σταματήσει αρκετές φορές σε οποιονδήποτε host, χωρίς να επηρεάζει τη λειτουργία του PVM ή οποιασδήποτε άλλης εφαρμογής στο σύστημα.

Οι λειτουργίες της pvm console είναι ευκολότερο να πραγματοποιούνται μέσω του XPVM, το οποίο αποτελεί το Graphical User Interface του PVM και είναι αρκετά φιλικό προς το χρήστη.

1.4 Ανάπτυξη εφαρμογών στο PVM

Η υπολογιστική φιλοσοφία του PVM έχει βασιστεί στο μοντέλο της ανταλλαγής μηνυμάτων, με αποτέλεσμα το PVM να είναι ένα ευέλικτο εργαλείο.
Υπάρχουν τρεις διαφορετικές βασικές μέθοδοι χρήσης του PVM :

- **transparent mode** κατά την οποία οι διαδικασίες εκτελούνται αυτόματα στο περισσότερο κατάλληλο μηχάνημα, όπου συνήθως είναι αυτό με το λιγότερο φόρτο εργασίας.

- **architecture - dependent mode** κατά την οποία ορισμένες διαδικασίες εκτελούνται σε μηχανήματα με αρχιτεκτονικές ορισμένες από το χρήστη.

- **low - level mode** κατά την οποία ο κάθε χρήστης ορίζει το συγκεκριμένο μηχάνημα, στο οποίο θα εκτελεστεί η κάθε διαδικασία.

Σε κάθε περίπτωση μπορεί να υπάρξει μια αλληλοεξάρτηση των διαδικασιών και πιο συγκεκριμένα παρέχεται η δυνατότητα να επικοινωνούν και να συγχρονίζονται μεταξύ τους. Έτσι η υπολογιστική μέθοδος του PVM το κατατάσσει στη κατηγορία των παράλληλων MIMD (Multiple Insfuction Multiple Data) συστημάτων.

Οι αναπτυσσόμενες εφαρμογές μπορούν να ακολουθήσουν τις τυπικές δομές των MIMD συστημάτων :

- **SIMD** (Single Insfuction Multiple Data) όπου οι παράλληλες διαδικασίες είναι ταυτόσημες.
- **Master - Slave** όπου κάποιες διαδικασίες ελέγχουν και κατευθύνουν τις υπόλοιπες.

Κατά την ανάπτυξη μιας εφαρμογής υπάρχουν ορισμένες παράμετροι, οι οποίες καθορίζουν την επίδοσή της. Οι παράμετρες αυτοί είναι σχετικές με τη μέθοδο υλοποίησης των διαδικασιών, το δίκτυο με το οποίο είναι συνδεμένοι οι hosts και η ισοκατανομή του φόρτου εργασίας.

1.4.1 Γενικές Αρχές Εφαρμογών Υψηλών Επιδόσεων

Με το PVM μπορούν να αναπτυχθούν πολύπλοκες και εξιδεικευμένες εφαρμογές. Για την υλοποίησή τους δεν υπάρχουν περιορισμοί και αυστηρά καθορισμένα πλαίσια, ώστε να δεσμεύεται ο χρήστης. Όμως, για να γίνει πλήρη εκμετάλλευση των παρεχόμενων δυνατοτήτων, είναι απαραίτητο κατά την υλοποίηση να λαμβάνει ο χρήστης υπ' όψη του τους ακόλουθους παράγοντες.

α. Ανταλλαγή μηνυμάτων

Καθοριστικός παράγοντας είναι το μέγεθος των ανταλλασσομένων μηνυμάτων. Η ανταλλαγή των μηνυμάτων μπορεί να γίνει με λίγα και μεγάλα μηνύματα ή με

περισσότερα και μικρότερα. Η καλύτερη επιλογή εξαρτάται από την εφαρμογή και το είδος της παράλληλης μηχανής που έχει δημιουργηθεί.

β. Μοντέλα παραλληλοποίησης

Οι εφαρμογές μπορεί να ακολουθούν δυο μοντέλα παραλληλοποίησης. Το πρώτο είναι σχετικό με την παραλληλοποίηση που προκύπτει από τα δεδομένα, ενώ το άλλο αναφέρεται στις αρχιτεκτονικές των υπολογιστικών συστημάτων που ταιριάζουν καλύτερα στα επί μέρους τμήματα της αναπτυσσόμενης εφαρμογής.
Ένας vector supercomputer είναι σε θέση να επιλύσει το τμήμα κάποιου προβλήματος που αναφέρεται σε διανύσματα, κάποιος multiprocessor μπορεί να λύσει το τμήμα στο οποίο αναφέρονται παράλληλοι υπολογισμοί, ενώ κάποιο graphics workstation μπορεί να χρησιμοποιηθεί για την γραφική παρουσίαση των παραγόμενων αποτελεσμάτων σε πραγματικό χρόνο.
Κάθε επιμέρους μηχάνημα εκτελεί διαφορετικού είδους διαδικασίες ίσως και με τα ίδια δεδομένα.

Στο μοντέλο της παραλληλοποίησης με βάση τα δεδομένα, αυτά διαμοιράζονται και προωθούνται στα επί μέρους υπολογιστικά συστήματα της παράλληλης μηχανής. Οι πράξεις τις περισσότερες φορές είναι παραπλήσιες και εκτελούνται για κάθε σύνολο δεδομένων σε κάθε υπολογιστικο σύστημα μέχρι να λυθεί το πρόβλημα.
Το είδος αυτό της παραλληλοποίησης είναι περισσότερο διαδεδομένο για distributed - memory multiprocessors, επειδή το ίδιο πρόγραμμα μπορεί να εκτελεστεί παράλληλα σε χιλιάδες επεξεργαστές.
Οι εφαρμογές που αναπτύσσονται συνήθως ανήκουν στο χώρο της Γραμμικής Άλγεβρας, των PDEs και γενικά αλγόριθμοι με πίνακες.

Στο PVM υπάρχει η δυνατότητα να αναπτυχθούν εφαρμογές, οι οποίες ανήκουν και στα δυο είδη μοντέλων, με την δημιουργία της κατάλληλης παράλληλης μηχανής.

1.4.2 Μεταφορά Δεδομένων

Από τα βασικότερα χαρακτηριστικά του PVM είναι η ανταλλαγή των μηνυμάτων ανάμεσα στα επιμέρους υπολογιστικά συστήματα, μέσω του δικτύου με το οποίο είναι συνδεμένα αυτά και την στιγμή που το δίκτυο εξυπηρετεί και τις ανάγκες των υπόλοιπων χρηστών. Έτσι αυτό το multiuser - multitasking περιβάλλον επιδρά στην λειτουργία της παράλληλης μηχανής κύρια με τους παρακάτω τρόπους.

α. Αρχιτεκτονικές Υπολογιστικών Συστημάτων

Η χρήση υπολογιστικών συστημάτων διαφορετικών αρχιτεκτονικών στη σύνθεση της PVM μηχανής μπορεί να επηρεάσει σημαντικά την απόδοση της εφαρμογής. Έτσι είναι απαραίτητο να υπάρχει σωστός καταμερισμός εργασίας, ώστε τα πιο αργά μηχανήματα να μην προκαλούν καθυστέρηση στην όλη διαδικασία. Υπάρχουν PVM διαδικασίες οι οποίες με την κατάλληλη χρήση προσδίδουν ένα δυναμικό χαρακτήρα στην ανάπτυξη της εφαρμογής.

β. Δυναμικότητα Δικτύων

Η υπολογιστική ισχύ και η ταχύτητα μεταφοράς δεδομένων του δικτύου είναι ασταθή μεγέθη, καθώς τα επηρεάζουν οι υπόλοιποι χρήστες του συστήματος. Έτσι μια εφαρμογή μπορεί να έχει πολύ καλή απόδοση κάποια ορισμένη χρονική στιγμή, ενώ μετά από λίγο μπορεί να έχει τελείως διαφορετική συμπεριφορά.

Το είδος και το μήκος του δικτύου αποτελούν καθοριστικό παράγοντα στη μεταφορά των μηνυμάτων. Αν για παράδειγμα ορισμένοι hosts βρίσκονται σε απομακρυσμένα μέρη, τότε είναι λογικό να λαμβάνεται υπό όψη η σχετική καθυστέρηση.

Για τις περιπτώσεις αργών ή υπερφορτωμένων δικτύων ο χρόνος της μεταφοράς των δεδομένων μπορεί να επηρεάσει σημαντικά την απόδοση του αλγορίθμου.

Όλα αυτά τα προβλήματα μπορούν να αντιμετωπιστούν με το συγχρονισμό των διαδικασιών, ώστε να αντιμετωπίζονται κατάλληλα οι πιθανές καθυστερήσεις που μπορούν να προκύψουν.

1.4.3 Ισοκατανομή Φόρτου Εργασίας

Σ' ένα multiuser network περιβάλλον η ισοκατονομή του φόρτου εργασίας έχει καθοριστικό ρόλο στην επίδοση μιας εφαρμογής. Τα μοντέλα, τα οποία σύντομα περιγράφονται παρακάτω, συνήθως εφαρμόζονται για παράλληλους αλγορίθμους.

- **Static Load Balancing.** Στη μέθοδο αυτή το αρχικό πρόβλημα διαμερίζεται σε άνισα υποπροβλήματα, τα οποία αναλαμβάνει αρχικά κάθε επεξεργαστής και μόνο μια φορά ανάλογα με την υπολογιστική του ισχύ. Όλες οι διαδικασίες ενεργοποιούνται και συνεργάζονται για την λύση του προβλήματος. Η μέθοδος αυτή είναι αποδοτική στην περίπτωση κατά την οποία τα επιμέρους μηχανήματα έχουν σταθερή υπολογιστική δύναμη και το δίκτυο επικοινωνίας δεν είναι ιδιαίτερα φορτωμένο.

- **Dynamic Load Balancing.** Για τις περιπτώσεις κατά τις οποίες, τόσο η υπολογιστική ικανότητα των υπολογιστικών συστημάτων, όσο και η κίνηση στο δίκτυο δεν είναι σε σταθερά πλαίσια, εφαρμόζεται η μέθοδος αυτή.

  Το περισσότερο χαρακτηριστικό παράδειγμα ονομάζεται **Pool of Tasks** και ακολουθεί το κλασσικό μοντέλο της **master / slave** αρχιτεκτονικής. Υπάρχουν δηλαδή κάποιες slave διαδικασίες, οι οποίες ελέγχονται πλήρως από μια master διαδικασία. Αρχικά προωθούνται οι slave διαδικασίες σε όλα τα υπολογιστικά συστήματα και μόλις ολοκληρωθεί κάποια από αυτές, αντικαθίστανται από κάποια άλλη και μέχρι την εξάντλησή τους.

  Η τεχνική αυτή είναι ιδιαίτερα αποδοτική σε περιπτώσεις που δεν απαιτείται η επικοινωνία μεταξύ slave διαδικασιών.

Το PVM διαθέτει ορισμένες διαδικασίες με τις οποίες είναι εφικτή η δημιουργία του δυναμικού χαρακτήρα της εφαρμογής. Έτσι επιτυγχάνεται η προσαρμογή της εφαρμογής σε κάθε δυνατή μεταβολή της συμπεριφοράς του συστήματος, χωρίς την επέμβαση του χρήστη.

1.5 Η Βιβλιοθήκη του PVM - libpvm

Το PVM3 υποστηρίζει τις δύο βασικότερες γλώσσες προγραμματισμού C και Fortran και είναι απαραίτητο τα προγράμματα στο στάδιο της μεταγλώττισης να χρησιμοποιούν την ειδική βιβλιοθήκη προγραμμάτων libpvm πριν από τη χρήση οποιασδήποτε άλλης.

Στη συνέχεια περιγράφουμε σύντομα τα υποπρογράμματα σε κατηγορίες ανάλογα με τη χρήση τους.

Αναλυτική παρουσίασή τους μπορεί να βρεθεί στο PVM Users' Guide.

1.5.1 Process Control

int tid = pvm_mytid(void) call pvmfmytid(tid)	Εγγραφή μιας διαδικασίας στο PVM
int info = pvm_exit(void) call pvmfexit(tid)	Διαγραφή μιας διαδικασίας από το PVM.
int numt = pvm_spawn(char *task, char **argv, int flag, char *where, int ntask, int *tids) call pvmfspawn(task, flag, where, ntask, tids, numt)	Ενεργοποίηση ntask αντιγράφων του εκτελέσιμου αρχείου task στο PVM
int info = pvm_kill(int tid) call pvmfkill(tid, info)	Σταμάτημα της διαδικασίας με αριθμό tid από το PVM

1.5.2 Dynamic Configuration

int info = pvm_addhosts(char **hosts, int nhost, int *infos) call pvmfaddhost(host, info)	Πρόσθεση ορισμένων hosts στη παράλληλη μηχανή.
int info = pvm_delhosts(char **hosts, int nhost, int *infos) call pvmfdelhost(host, info)	Αφαίρεση ορισμένων hosts από τη παράλληλη μηχανή.

1.5.3 Signaling

int info = pvm_sendsig(int tid, int signum) call pvmfsendsig(tid, signum, info)	Μετάδοση του μηνύματος **signum** στη διαδικασία **tid**.
int info = pvm_notify(int what, int msgtag, int cnt, int tids) call pvmfnotify(what, msgtag, cnt, tids, info)	Ενημέρωση για διαδικασίες ή κατάσταση ορισμένων hosts στη παράλληλη μηχανή.

1.5.4 Information

int tid = pvm_parent(void) call pvmfparent(tid)	Επιστροφή του αριθμού tid της διαδικασίας που ενεργοποιήθηκε
int pstat = pvm_pstat(int tid) call pvmfpstat(tid, pstat)	Πληροφορίες για τη κατάσταση της διαδικασίας με αριθμό tid
int pstat = pvm_mstat(char *host) call pvmfmstat(host, mstat)	Πληροφορίες για τη κατάσταση κάποιου host
int info = pvm_config(int *nhost, int *narch, struct pvmhostinfo **hostp) call pvmfconfig(nhost, narch, dtid, name, arch, speed, info)	Πληροφορίες για τη κατάσταση της παράλληλης μηχανής με διαφορετικούς hosts αριθμού **nhost.**
int info = pvm_tasks(int wich, int *ntask, struct pvmtaskinfo **taskp) call pvmftasks(which, ntask, tid, ptid, dtid, flag, aout, info)	Πληροφορίες για τη κατάσταση των διαδικασιών **which** που τρέχουν στη παράλληλη μηχανή.
int dtid = pvm_tidtohost(int tid) call pvmftidtohost(tid, dtit)	Πληροφορίες για το host που τρέχει μια task.

1.5.5 Dynamic Process Groups

int inum = pvm_joingroup(char *group) call pvmfjoingroup(group, inum)	Ενσωμάτωση μιας διαδικασίας στην ομάδα **group**
int info = pvm_lvgroup(char *group) call pvmflvgroup(group, info)	Απομάκρυνση μιας διαδικασίας από την ομάδα **group**
int tid = pvm_gettid(char *group, int inum) call pvmfgettid(group, inum, tid)	Επιστροφή μιας διαδικασίας από την ομάδα της.
int inum = pvm_getinst(char *group, int tid) call pvmfgetint(group, tid, inum)	Επιστροφή του στιγμιαίου αριθμού μιας διαδικασίας από μια ομάδα.
int size = pvm_gsize(char *group) call pvmfgsize(group, size)	Επιστρέφει τον αριθμό των μελών μιας ομάδας.
int info = pvm_barrier(char *group, int count) call pvmfbarrier(group, count, info)	Διακοπή μιας διαδικασίας μέχρι να την καλέσουν **count** μέλη της ομάδας.
int info = pvm_bcast(char *group, int msgtag) call pvmfbcast(group, msgtag, info)	Μετάδοση μηνύματος στα υπόλοιπα μέλη της ομάδας.
int info = pvm_reduce(void (*func)(), void *data, int nitem, int datatype, int msgtag, char *group, int root) call pvmfreduce(func, data, count, datatype, msgtag, group, root, info)	Αριθμητικές πράξεις σε όλα τα μέλη μιας συγκεκριμένης ομάδας, όπως καθολική άθροιση ή μέγιστο.

1.5.6 Setting and Getting Options

int oldval = pvm_setopt(int what, int val) call pvmfsetopt(what, val, oldval)	Ορισμός ιδιοτήτων της παράλληλης μηχανής.
int val = pvm_getopt(int what) call pvmfgetopt(what, val)	Εφαρμογή ορισμένων ιδιο- τήτων στη παράλληλη μηχανή.

1.5.7 Message Passing

α. Message Buffers

int bufid = pvm_mkbuf(int encoding) call pvmfmkbuf(encoding, bufid)	Δημιουργία buffer και καθο- ρισμός μεθόδου μετάδοσης.
int bufid = pvm_initsend(int encoding) call pvmfinitsend(encoding, bufid)	Καθαρισμός και προετοιμασία του buffer για μετάδοση.
int info = pvm_freebuf(int bufid) call pvmffreebuf(bufid, info)	Αποενεργοποίηση ενός buffer.
int bufid = pvm_getsbuf(void) call pvmfgetsbuf(bufid)	Επιστροφή αριθμού ενός ενεργού buffer μετάδοσης.
int bufid = pvm_getrbuf(void) call pvmfgetrbuf(bufid)	Επιστροφή αριθμού ενός ενεργού buffer λήψης.
int oldbuf = pvm_setsbuf(int bufid) call pvmfsetsbuf(bufid, oldbuf)	Επαναφορά ενός buffer μετάδοσης σε προηγούμενη κατάσταση.
int oldbuf = pvm_setrbuf(int bufid) call pvmfsetrbuf(bufid, oldbuf)	Επαναφορά ενός buffer λήψης σε προηγούμενη κατάσταση.

β. Packing Data

int info = pvm_packf(const char *fmt, ...) call pvmfpack(what, xp, nitem, stride, info)	Προετοιμασία δεδομένων για μεταφορά.

γ. Unpacking Data

int info = pvm_unpackf(const char *fmt, ...) call pvmfunpack(what, xp, nitem, stride, info)	Επαναφορά δεδομένων μετά τη μεταφορά.

δ. Sending and Receiving Data

int info = pvm_send(int tid, int msgtag) call pvmfsend(tid, msgtag, info)	Αρίθμηση μηνύματος και μεταφορά του.
int info = pvm_mcast(int *tids, int ntask, int msgtag) call pvmfcast(ntask, tids, msgtag, info)	Μεταφορά μηνύματος σε καθορισμένες διαδικασίες.
int info = pvm_psend(int tid, int msgtag, void *vp, int cnt, int type) call pvmfpsend(tid, msgtag, xp, cnt, type, info)	Προετοιμασία και μεταφορά των δεδομένων ενός πίνακα.
int bufid = pvm_recv(int tid, int msgtag) call pvmfrecv(tid, msgtag, bufid)	Λήψη μηνύματος από καθορισμένη διαδικασία.
int bufid = pvm_nrecv(int tid, int msgtag) call pvmfnrecv(tid, msgtag, bufid)	Ελεγχος και λήψη μηνύματος από καθορισμένη διαδικασία.
int bufid = pvm_probe(int tid, int msgtag) call pvmfprobe(tid, msgtag, bufid)	Ελεγχος για λήψη μηνύματος από καθορισμένη διαδικασία.
int info = pvm_bufinfo(int bufid, int *bytes, int *msgtag, int *tid) call pvmfbufinfo(bufid, bytes, msgtag, tid, info)	Πληροφορίες σχετικές με μηνύματα πριν από τη λήψη τους.
int bufid = pvm_trecv(int tid, int msgtag,) struct timeval *tmout) call pvmftrecv(tid, msgtag, sec, usec, bufid)	Λήψη μηνυμάτων μέσα σε προκαθορισμένα χρονικά πλαίσια.
int info = pvm_precv(int tid, int msgtag, void *vp, int cnt, int type, int *rtid, int *rtag, int *rcnt) call pvmfprecv(tid, msgtag, xp, cnt, type, rtid, rtag, rcnt, info)	Λήψη και επαναφορά μετά η μεταφορά των δεδομένων ενός μηνύματος.

1.6 XPVM

Το XPVM αποτελεί το Graphical User Interface του PVM και προσφέρει πλήρη έλεγχο της παράλληλης μηχανής, γραφική παρουσίαση της κατάστασης κάθε host, λεπτομερή απεικόνιση όλων των σταδίων εκτέλεσης μιας εφαρμογής και άλλες χρήσιμες πληροφορίες σχετικές με επιδόσεις και βελτιστοποίηση μεγεθών.

Το XPVM συνδυάζει τις λειτουργίες και τις δυνατότητες της pvm console μαζί με την γραφική παρουσίαση της εκτέλεσης της εφαρμογής. Είναι μια X-window εφαρμογή εύχρηστη και φιλική προς το χρήστη, η οποία προσφέρεται με τον ίδιο ακριβώς τρόπο με το PVM μέσω **netlib** από το directory **pvm3/xpvm**.

Τα ανάλογα εκτελέσιμα αρχεία προσφέρονται έτοιμα για τα περισσότερο διαδεδομένα μηχανήματα, ενώ είναι διαθέσιμα και τα C-sourse αρχεία.

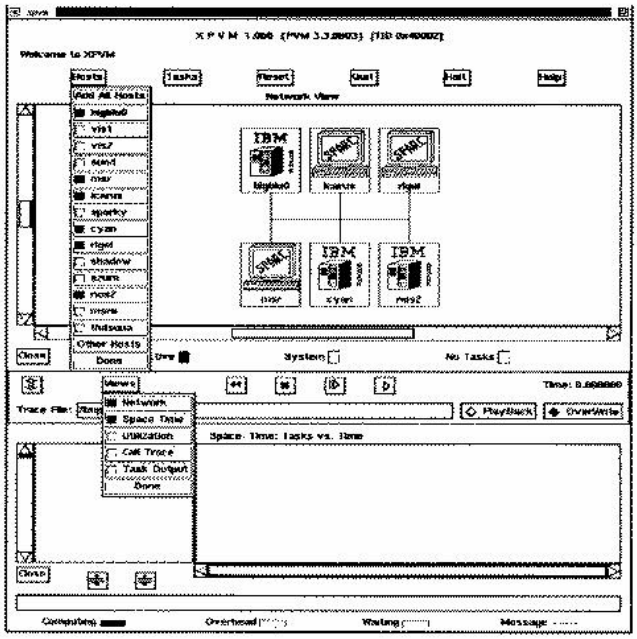
Το XPVM χρησιμοποιεί το λογισμικό TCL/TK, οπότε για την περίπτωση της δημιουργίας εκτελέσιμων είναι απαραίτητη η απόκτησή του με anonymous ftp από $sprite.berkley.edu$.

Ο χρήστης θα πρέπει να ορίσει στο περιβάλλον του δύο νέες μεταβλητές, τις TK_LIBRARY και TCL_LIBRARY, σύμφωνα με τη θέση στο σύστημα των βιβλιοθηκών αυτών.

Το XPVM χρησιμοποιεί δικά του hostfiles, τα οποία αναζητά με την ονομασία **$HOME/.xpvm_hosts** και για την περίπτωση κατά την οποία δεν τα έχει δημιουργήσει ο χρήστης, τότε απλά ξεκινάει το PVM μόνο σε αυτό το μηχάνημα.

Κατά την έναρξη της λειτουργίας του XPVM δημιουργείται η παράλληλη μηχανή με τα χαρακτηριστικά που περιγράφονται στο .xpvm_hosts αρχείο, ενεργοποιώντας τους pvm daemons σε κάθε host, όταν αυτοί δεν είναι ήδη ενεργοί.

Η πρώτη επιλογή πάνω αριστερά με την ένδειξη **Hosts** στο XPVM παράθυρο, επιτρέπει την εισαγωγή ή την απομάκρυνση διαφορων hosts από την παράλληλη μηχανή. Η διαδικασία αυτή εμφανίζεται στην Εικόνα 2.



Εικόνα 2 : Πρόσθεση - Απομάκρυνση hosts μέσω XPVM

Οι επιλογές της πάνω πλευράς είναι αφιερωμένες στη λειτουργία της PVM console, ενώ ο χώρος ακριβώς από πάνω τους χρησιμεύει για την παρουσίαση των μηνυμάτων.

Με την επιλογή Tasks μπορούν να γίνουν όλες οι ενέργειες σχετικές με την εκτέλεση εφαρμογών. Σε αυτό το σημείο θα πρέπει να επισημάνουμε ότι, βασική προϋπόθεση είναι η ύπαρξη των ανάλογων εκτελέσιμων αρχείων της εφαρμογής στη θέση που έχει οριστεί. Με άλλα λόγια θα πρέπει να έχει γίνει μεταγλώττιση των κατάλληλων τμημάτων του προγράμματος σε κάθε host και τα εκτελέσιμα να έχουν μεταφερθεί στο ήδη ορισμένο σημείο, συνήθως στο directory $\$PVM\_ROOT/bin/\$PVM\_ARCH$.

Η εντολή **Spawn** ενεργοποιεί την εφαρμογή. Υπάρχουν αρκετές επιλογές για την υλοποίηση αυτής της εντολής, ενώ παράλληλα προσφέρεται η δυνατότητα

της ταυτόχρονης πολλαπλής ενεργοποίησης της εφαρμογής. Έτσι μπορεί να ενεργοποιηθεί η εφαρμογή σε κάποιο συγκεκριμένο host ή σε μια ορισμένη αρχιτεκτονική. Ακόμα είναι εφικτή η επιλογή ανάμεσα σε τέσσερις παραμέτρους εκτέλεσης της εφαρμογής, οι οποίες αναφέρονται σε πληροφορίες που συλλέγονται κατά τη διάρκεια της εκτέλεσης.

Εκτός από την επιλογή **Spawn** υπάρχουν και άλλες τρεις επιλογές σχετικές με την διαχείρηση διαδικασιών. Η επιλογή **Kill** διακόπτει κάποια tasks, ενώ οι άλλες δύο σχετίζονται με την επικοινωνία ανάμεσά τους.

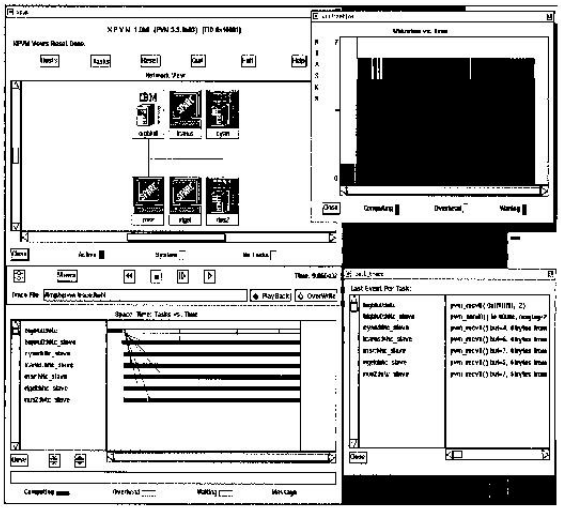
Δίπλα από την επιλογή **Tasks** υπάρχει η επιλογή **Quit** η οποία προσφέρει έξοδο από το πρόγραμμα, χωρίς να σταματάει τους pvm deamons. Αυτό μπορεί να επιτευχθεί με την διπλανή επιλογή **Halt**.

Η τελευταία επιλογή σ' αυτό το τομέα είναι η **Help**, η οποία παρέχει on line βοήθεια και επεξήγηση κάθε επιλογής.

Ακριβώς από κάτω υπάρχει η δυνατότητα της γραφικής παρουσίασης της παράλληλης μηχανής με διαφορετικό εικονίδιο για κάθε αρχιτεκτονική. Κατά την ενεργοποίηση της εφαρμογής παρουσιάζονται με λευκό χρώμα οι hosts που δεν εκτελούν tasks, ενώ με κίτρινο και πράσινο εμφανίζονται εκείνοι που εκτελούν tasks του συστήματος και της εφαρμογής αντίστοιχα.

Στη συνέχεια υπάρχουν επιλογές σχετικές με την παρουσίαση ορισμένων στοιχείων από την ενεργοποίηση μιας εφαρμογής. Στο δεξιό σημείο εμανίζονται οι επιλογές που επιτρέπουν την λειτουργία ενός είδους κινηματογραφικής συσκευής, με την οποία είναι δυνατή η επαναπαρουσίαση της εκτέλεσης μιας εφαρμογής. Στο δεξιό σημείο εμφανίζεται ο χρόνος εκτέλεσης, ενώ στο αριστερό υπάρχει η επιλογή **Views** με πέντε υποεπιλογές. Οι υποεπιλογές αυτές καθορίζουν το είδος των στοιχείων που παρουσιάζονται στο χρήστη.

Η πρώτη επιλογή **Network** παρουσιάζει τους hosts και το δίκτυο, όπως αναφέρθηκε παραπάνω και ακολουθεί η επιλογή **Space - Time**. Με την επιλογή



Εικόνα 3 : Utilization μέσω XPVM

αυτή επιτρέπεται η παρουσίαση των tasks στο χρόνο εκτέλεσης. Αυτό γίνεται ακριβώς από κάτω, όπου αριστερά εμφανίζεται το όνομα του task και του host και δεξιότερα σε ρυθμιζόμενες οριζόντιες λωρίδες οι tasks στο χρόνο. Για την περίπτωση κατά την οποία γίνεται υπολογισμός εμφανίζεται η task με πράσινη απόχρωση, με κίτρινη στις περιπτώσεις overhead και λευκό για τις περιπτώσεις αναμονής δεδομένων από άλλη task. Με κόκινες γραμμές παρουσιάζεται η μεταφορά δεδομένων ανάμεσα στις tasks.

Η επόμενη επιλογή **Utilization** παρουσιάζει την εκμετάλλευση της υπολογιστικής ισχύς της παράλληλης μηχανής. Σ' ένα καινούριο παράθυρο, όπως φαίνεται στην Εικόνα 3, εμφανίζονται οι καθυστερήσεις και το overhead με διαφορετικά χρώματα κι έτσι είναι πολύ εύκολο να καταλάβει ο χρήστης κατά πόσο η υλοποίηση που έχει γίνει αξιοποιεί τις δυνατότητες που παρέχονται από το PVM.

Η επιλογή **Call Trace** αναφέρεται στο αρχείο, το οποίο χρησιμοποιείται για την αποθήκευση στοιχείων, ώστε να είναι εφικτή η επαναπαρουσίαση της

εκτέλεσης μιας εφαρμογής.

Οι διαδικασίες χρησιμοποιούν κάποια αρχεία για να εκτυπώσουν μηνύματα, τα οποία συνήθως δημιουργούνται στο directory **/tmp**. Με την τελευταία επιλογή **Task Output** είναι δυνατή η εμφάνιση των μηνυμάτων των διαδικασιών σε ένα ειδικό παράθυρο, το οποίο δημιουργείται μετά την επιλογή.

Το XPVM είναι ένα πολύ χρήσιμο εργαλείο, το οποίο καλύπτει το κενό της φιλικότητας προς το χρήστη και της ταυτόχρονης παρουσίασης πλήθους στοιχείων που προκύπτουν από την ανάπτυξη κάποιας εφαρμογής μέσω του PVM.

1.7 HeNCE

Το HeNCE - Heterogeneous Network Computing Environment - είναι ένα γραφικό περιβάλλον για παράλληλο προγραμματισμό. Ακήκει στη κατηγορία των πολύ φιλικών προς το χρήστη X-window εφαρμογών και παρέχει τη δυνατότητα της δημιουργίας, μεταγλώττισης, εκτέλεσης και διόρθωσης παράλληλων προγραμμάτων, τα οποία προορίζονται για το σύστημα PVM.

Με το HeNCE ένας ερευνητής χωρίς εξοικείωση σε παράλληλο προγραμματισμό μπορεί να συνδυάσει σειριακά υποπρογράμματα και να δημιουργήσει ένα παράλληλο πρόγραμμα, το οποίο μπορεί να εκτελεστεί σε μια συλλογή υπολογιστικών συστημάτων με διαφορετικές αρχιτεκτονικές.

Ο χρήστης δημιουργεί το παράλληλο πρόγραμμα σχεδιάζοντας ένα γράφημα, το οποίο έμμεσα απεικονίζει τους παράλληλους υπολογισμούς. Στη δημιουργία των προγραμμάτων αυτών δεν υπάρχουν συγκεκριμένες τεχνικές και παραδοσιακοί κανόνες, οι οποίοι να δεσμεύουν το χρήστη.

Μετά από τη σχεδίαση και την ολοκλήρωση της διαδικασίας της δημιουργίας του παράλληλου προγράμματος, το HeNCE αυτόματα το μεταγλωττίζει και δημιουργεί το εκτελέσιμο παράλληλο πρόγραμμα για το PVM.

Το γράφημα, μέσω του οποίου απεικονίζεται το παράλληλο πρόγραμμα, είναι κατευθυνόμενο γράφημα και οι κόμβοι του αντιστοιχούν σε ανάλογα καλέσματα υποπρογραμμάτων. Αυτά τα υποπρογράμματα μπορεί να είναι συμβατικές σειριακές υπορουτίνες γραμμένες σε C ή Fortran. Χρησιμοποιούνται τόξα για να καθοριστεί η κατεύθυνση των υπολογισμών και κάθε κόμβος ενεργοποιεί το υποπρόγραμμά του μόνο για την περίπτωση όπου οι προηγούμενοι κόμβοι έχουν εκτελέσει τα δικά τους.

1.7.1 Απόκτηση - Εγκατάσταση

Η απόκτηση του λογισμικού πακέτου HeNCE είναι εφικτή για κάθε χρήστη του Internet, εφόσων προσφέρεται ελεύθερα, όπως και το PVM από το server **netlib**. Ο τρόπος μπορεί να είναι ένας από τους ακόλουθους :

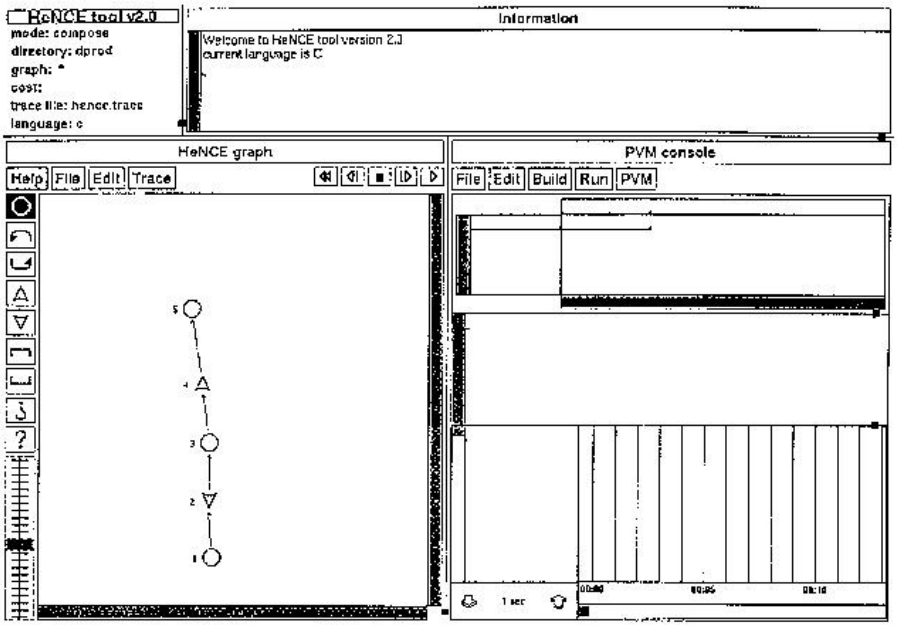
- Μέσω του προγράμματος **xnetlib** από το directory hence.
- Με anonymous ftp στη διεύθυνση **ftp.netlib.org** [128.169.92.17]. Τα σχετικά αρχεία περιέχονται στο directory hence.
- Στέλνοντας e-mail στη διεύθυνση netlib@ornl.gov με περιεχόμενο το παρακάτω μήνυμα : **send index from hence** . Αυτόματα ο χρήστης θα λάβει ένα αρχείο με τις λίστες των σχετικών αρχείων και τις οδηγίες απόκτησης τους, χρησιμοποιώντας το ηλεκτρονικό ταχυδρομείο.
- Μέσω του World Wide Web server της Netlib με διεύθυνση **http://www.netlib.org/hence/index.html**.

Το λογισμικό λαμβάνεται σε συμπιεσμένη μορφή και είναι διαθέσιμα τα κατάλληλα εκτελέσιμα για τους πιο διαδεδομένους τύπους μηχανημάτων, ενώ ταυτόχρονα προσφέρεται και το C source αρχείο.

Η εγκατάσταση του HeNCE προϋποθέτει τη προηγούμενη σωστή εγκατάσταση του PVM. Είναι απαραίτητο να έχει οριστεί σωστά στο περιβάλλον του χρήστη η μεταβλητή **PVM_ROOT**.

Το βασικό εκτελέσιμο πρόγραμμα έχει την ονομασία **htool** και χρησιμοποιεί σαν αρχείο εισόδου δεδομένων το .htool, το οποίο θα πρέπει να βρίσκεται στο home directory του χρήστη. Στο αρχείο αυτό μπορούν να οριστούν διάφοροι παράμετροι όπως τα χρώματα, επιλεγμένες θέσεις διάφορων αρχείων και γενικά να γίνουν κάθε είδους ρυθμίσεις.

Στην Εικόνα 4 εμφανίζεται ένα παράδειγμα λειτουργίας του htool.

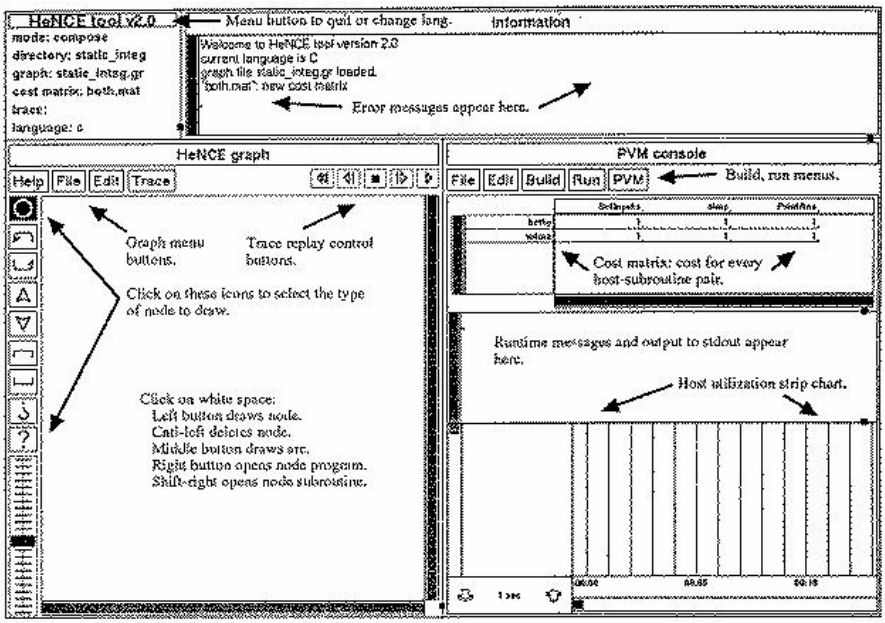


Εικόνα 4 : Htool σε λειτουργία.

1.7.2 Ανάπτυξη Εφαρμογών στο HeNCE

Το Graphical User Interface του HeNCE καλείται htool και χρησιμοποιεί το περιβάλλον των X-windows σε έκδοση X11R4 ή νεότερη σε ένα workstation. Η μέθοδος ανάπτυξης εφαρμογών στο HeNCE αποτελείται από τα παρακάτω βήματα :

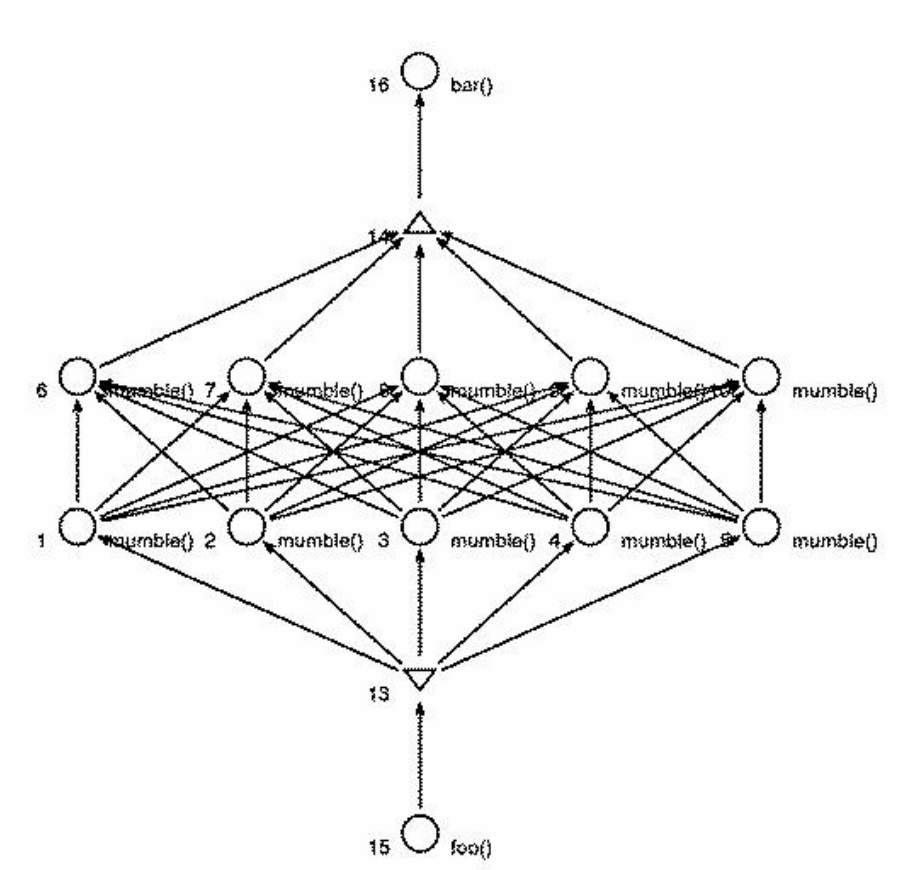
- Σχεδιασμός του κατευθυνόμενου γραφήματος που περιγράφει έμμεσα τη παράλληλη δομή της εφαρμογής.
- Καθορισμός των υποπρογραμμάτων που αντιστοιχούν σε κάθε κόμβο του γραφήματος.
- Ορισμός σειριακών υποπρογραμμάτων σε κάθε επεξεργαστή.



Εικόνα 5 : Η X-window εφαρμογή htool.

- Αντιστόιχιση υποπρογραμμάτων με τα επιμέρους υπολογιστικά συστήματα της παράλληλης μηχανής με τη δημιουργία του πίνακα κόστους.
- Μεταγλώττιση υποπρογραμμάτων και αυτόματη δημιουργία του κυρίου προγράμματος από το htool.
- Εκτέλεση της εφαρμογής στο παράλληλο σύστημα.
- Επαναληπτική παρουσίαση της προηγούμενης εκτέλεσης και μελέτη από τα γραφήματα απόδοσης.

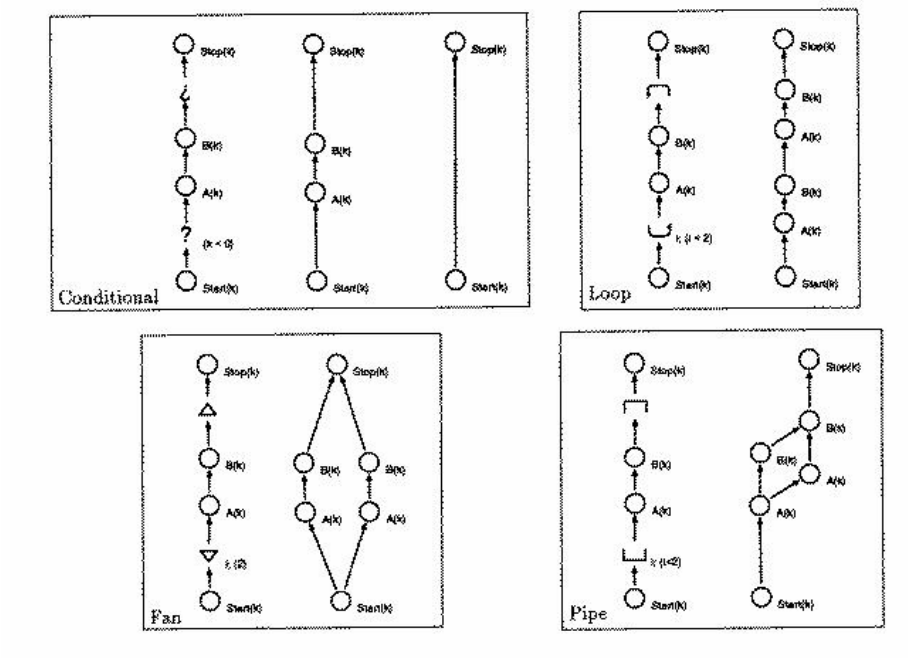
Στην Εικόνα 5 παρουσιάζονται τα επιμέρους τμήματα της X-window εφαρμογής htool, στα οποία υλοποιούνται τα παραπάνω βήματα. Έτσι στο πάνω αριστερό άκρο υπάρχουν οι επιλογές για αλλαγή directory, γλώσσας προγραμματισμού και έξοδος προγράμματος. Ο χώρος δίπλα είναι αφιερωμένος για εκτύπωση μηνυμάτων.



Εικόνα 6 : Κατευθυνόμενο γράφημα στο HeNCE.

Ο σχεδιασμός του γραφήματος γίνεται στο κάτω και αριστερό τμήμα του παραθύρου. Στην πάνω πλευρά υπάρχουν τέσσερις επιλογές σχετικές με την επεξεργασία γραφημάτων. Η πρώτη αναφέρεται σε on line βοήθεια, η δεύτερη σε επεξεργασία αρχείων, η τρίτη σε δημιουργία γραφημάτων και η τελευταία σε επαναπαρουσίαση προηγούμενων εκτελέσεων εφαρμογών. Στην Εικόνα 6 εμφανίζεται ένα παράδειγμα κατευθυνόμενου γραφήματος.

Στην αριστερή πλευρά υπάρχουν όλες οι δυνατές επιλογές κόμβων του γραφήματος. Η Εικόνα 7 εμφανίζει τα βασικά είδη κόμβων για διάφορες εφαρμογές. Αφού έχει γίνει η κατάλληλη επιλογή του κόμβου, μπορεί να σχεδιαστεί με το αριστερό πλήκτρο του ποντικιού. Πατώντας επιπλέον το πλήκτρο control γίνεται διαγραφή του κόμβου, ενώ με το shift επιτυγχάνεται η μετατόπισή του.

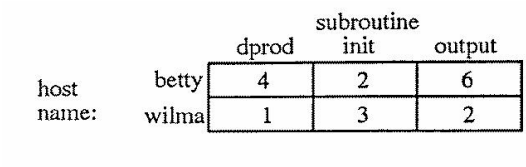


Εικόνα 7 : Τα βασικά είδη κόμβων και εφαρμογές τους.

Έχοντας πατημένο το μεσαίο πλήκτρο του ποντικιού σχεδιάζεται ένα βέλος, το οποίο ενώνει δυο κόμβους και καθορίζει την κατεύθυνση των υπολογισμών. Με παράλληλο πάτημα του πλήκτρου control γίνεται διαγραφή τόξου, ενώ με το shift επιτυγχάνεται μετατόπιση.

Μετά την ολοκλήρωση της διαδικασίας του σχεδιασμού του κατευθυνόμενου γραφήματος, ακολουθεί η αντιστοίχιση των κόμβων με τα σειριακά υποπρογράμματα. Αυτό επιτυγχάνεται με το ταυτόχρονο πάτημα του δεξιού πλήκτρου του ποντικιού μαζί με το shift. Το υποπρόγραμμα γράφεται σε ένα επιπλέον ενεργοποιημένο παράθυρο και σε αρχείο στο tmp directory.

Τέλος γίνεται η δημιουργία του προγράμματος κάθε κόμβου με το πάτημα του δεξιού πλήκτρου του ποντικιού. Το πρόγραμμα αυτό είναι διαφορετικό για κάθε είδους κόμβων και ουσιαστικά καθορίζει τις μεταβλητές που χρησιμοποιεί το υποπρόγραμμα του κόμβου. Οι πληροφορίες αυτές χρησιμοποιούνται από το σύστημα για την αυτόματη δημιουργία του κύριου προγράμματος.



Εικόνα 8 : Πίνακας Κόστους.

Το επόμενο βήμα είναι ο καθορισμός των hosts, οι οποίοι θα αναλάβουν την εκτέλεση των κατάλληλων σειριακών υποπρογραμμάτων. Αυτό επιτυγχάνεται με την δημιουργία του Πίνακα Κόστους, στον οποίο ορίζονται επίσης και τα βάρη με τα οποία θα γίνει η εκτέλεση αυτή.

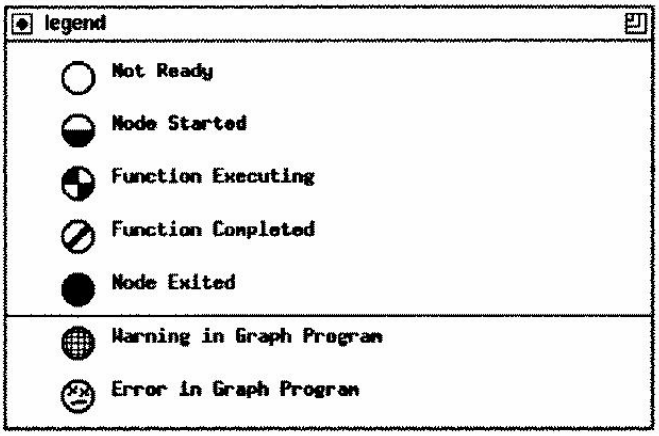
Η Εικόνα 8 εμφανίζει έναν τέτοιο πίνακα. Στην πρώτη γραμμή εισάγονται τα ονόματα των υποπρογραμμάτων και στην πρώτη στήλη αυτά των hosts. Ο χρήστης συμπληρώνει με μη μηδενικούς φυσικούς αριθμούς τις κενές θέσεις, καθορίζοντας έτσι το κόστος εκτέλεσης κάθε υποπρογράμματος σε κάθε επιμέρους υπολογιστικό σύστημα της παράλληλης μηχανής.

Στο παράθυρο του htool οι πίνακες κόστους δημιουργούνται στο μέσο δεξιό τμήμα. Ακριβώς από πάνω υπάρχουν πέντε επιλογές σχετικές με τη κατασκευή του πίνακα και τη λειτουργία του PVM συστήματος.

Οι δυό πρώτες επιλογές επιτρέπουν τη διαχείρηση των αρχείων των πινάκων και την κατασκευή τους. Η επιλογή **Build** επιτρέπει τη μεταγλώττιση της αναπτυσόμενης εφαρμογής, ενώ με την επιλογή **Run** επιτυγχάνεται η εκτέλεση ή η αναστολή της.

Στην διπλανή επιλογή **PVM** υπάρχουν διάφορες υποεπιλογές που ελέγχουν και ενημερώνουν το χρήστη για την κατάσταση της παράλληλης μηχανής.

Το htool δεν είναι σε θέση να δημιουργήσει την παράλληλη μηχανή. Ο χρήστης έχει την υποχρέωση να ενεργοποιήσει τους pvm daemons στα επιμέρους υπολογιστικά συστήματα εξωτερικά.



Εικόνα 9 : Τύποι υπολογιστικών κόμβων μετά την εκτέλεση της εφαρμογής.

Κατά τη διάρκεια της εκτέλεσης μιας εφαρμογής μέσω του HeNCE, παρουσιάζονται τα διάφορα στάδια υπολογισμών με διαδοχικές εναλλαγές των κόμβων του γραφήματος. Στην Εικόνα 9 εμφανίζονται οι διάφοροι τύποι κόμβων για κάθε περίπτωση.

Ταυτόχρονα με την γραφική παρουσίαση της εκτέλεσης της εφαρμογής πάνω στο γράφημα, υπάρχει και η απεικόνιση του χρόνου. Αυτό γίνεται στο κάτω δεξί τμήμα του παραθύρου κι έτσι ο χρήστης μπορεί να έχει πλήρη ενημέρωση για τη χρονική εξέλιξη της εκτέλεσης. Ακόμα παρέχεται η δυνατότητα του ορισμού της κλίμακας του χρόνου για λεπτομερή μελέτη και πληρέστερη κατανόηση των γεγονότων που συμβαίνουν.

Στη διαδικασία εκτέλεσης μιας εφαρμογής, δημιουργείται ένα trace αρχείο, το οποίο αποθηκεύει πληροφορίες, ώστε να είναι δυνατή η επαναπαρουσίασή της. Η δυνατότητα αυτή συμβάλλει στην πληρέστερη κατανόηση της ανάπτυξης της εφαρμογής. Ο έλεγχος της διαδικασίας αυτής γίνεται μέσω της επιλογής **trace** και των επιλογών που υπάρχουν ακριβώς δίπλα της.

Υπάρχει η δυνατότητα της επικοινωνίας με τους δημιουργούς του HeNCE μέσω e-mail στη διεύθυνση **hence@cs.utk.edu** για κάθε σχετική πληροφορία.

2 ΜΕΘΟΔΟΙ ΑΡΙΘΜΗΤΙΚΗΣ ΕΠΙΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΩΝ ΣΥΝΟΡΙΑΚΩΝ ΤΙΜΩΝ ΕΛΛΕΙΠΤΙΚΟΥ ΤΥΠΟΥ

2.1 Εισαγωγή

Είναι γνωστό ότι η γενική μορφή Προβλημάτων Συνοριακών Τιμών (ΠΣΤ) περιγράφεται από τις σχέσεις

$$(\Pi\Sigma\mathrm{T}) \quad \begin{cases} \mathcal{L}u(x,y) \;=\; f(x,y) \;\;,\;\; (x,y) \in \Omega \\ \\ \mathcal{B}u(x,y) \;=\; g(x,y) \;\;,\;\; (x,y) \in \partial\Omega \end{cases} \tag{1}$$

όπου οι διαφορετικοί τελεστές $\mathcal{L}$ και $\mathcal{B}$ ορίζονται αντίστοιχα σαν

$$\begin{cases} \mathcal{L} \;\equiv\; a(x,y)\frac{\partial^2}{\partial x^2} + 2b(x,y)\frac{\partial^2}{\partial x \partial y} + c(x,y)\frac{\partial^2}{\partial y^2} + d(x,y)\frac{\partial}{\partial x} + e(x,y)\frac{\partial}{\partial y} + h(x,y) \\ \\ \mathcal{B} \;\equiv\; \alpha(x,y) + \beta(x,y)\frac{\partial}{\partial \vec{n}} \end{cases} \tag{2}$$

Η συνθήκη $a(x,y)c(x,y) > b^2(x,y)$ χαρακτηρίζει τον **ελλειπτικό** τύπο του προβλήματος και συνεπάγεται ότι οι συναρτήσεις $a(x,y)$ και $c(x,y)$ είναι ομόσημες και μη μηδενικές.

Από τα κλασικότερα παραδείγματα ελλειπτικών ΠΣΤ είναι και το Poisson-Dirichlet πρόβλημα που ορίζεται σαν

$$\begin{cases} \nabla^2 u(x,y) \;\;=\;\; f(x,y) \;\;,\;\; (x,y) \in \Omega \equiv [0,1] \times [0,1] \\ \\ \quad u(x,y) \;\;=\;\; g(x,y) \;\;,\;\; (x,y) \in \partial\Omega \end{cases} \tag{3}$$

Το Poisson-Dirichlet πρόβλημα αναφέρεται στην βιβλιογραφία ως το πρόβλημα μοντέλο ελλειπτικών ΠΣΤ για την ανάπτυξη και μελέτη αριθμητικών μεθόδων και ως τέτοιο θα χρησιμοποιηθεί στην παρούσα μελέτη.

Ανάμεσα στις περισσότερο διαδεδομένες μεθόδους αριθμητικής επίλυσης ΠΣΤ είναι και αυτές των Πεπερασμένων Στοιχείων, οι οποίες περιγράφονται σχηματικά με τα παρακάτω βήματα :

- **Βήμα 0:** Γεωμετρική Διαμέριση του πεδίου Ω σε πεπερασμένο πλήθος στοιχείων.

- **Βήμα 1:** Επιλογή n γραμμικά ανεξάρτητων κατά τμήματα πολυωνυμικών συναρτήσεων $\Phi_1, \ldots, \Phi_n$, οι οποίες ονομάζονται συναρτήσεις **βάσης** και χρησιμοποιούνται στη προσέγγιση της πραγματικής λύσης $u(x, y)$ του ΠΣΤ ως εξής

$$u(x,y) \simeq u_n(x,y) = a_1\Phi_1(x,y) + \ldots + a_n\Phi_n(x,y) = \sum_{k=1}^{n} a_k\Phi_k(x,y). \quad (4)$$

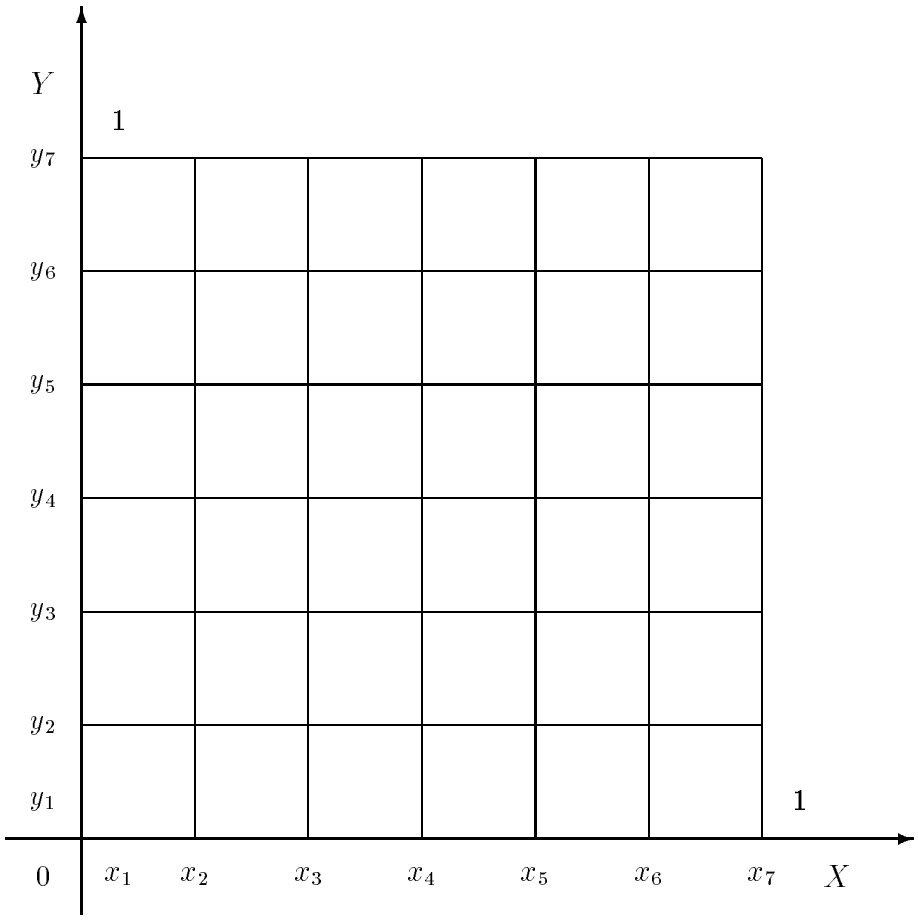
- **Βήμα 2:** Επιλογή μεθόδου διακριτοποίησης (Galerkin, Rayleigh-Ritz, Least Squares, Collocation) για μετάβαση από το **συνεχή** χώρο στο **διακριτό** δηλαδή μετατροπή του ΠΣΤ σε ένα γραμμικό σύστημα

$$C\vec{\mathbf{a}} = \vec{\mathbf{b}} \quad (5)$$

όπου $C \in \mathbb{R}^{n,n}$, $\vec{\mathbf{a}} = [a_1 \;\; a_2 \;\; \cdots \;\; a_n]^T$ και $\vec{\mathbf{b}} = [b_1 \;\; b_2 \;\; \cdots \;\; b_n]^T$.

- **Βήμα 3:** Επιλογή μεθόδου για την επίλυση του γραμμικού συστήματος και προσδιορισμός των αγνώστων $a_1, \ldots, a_n$. Η επιλογή αυτή βασίζεται στο μέγεθος και στις ιδιότητες του παραγόμενου γραμμικού συστήματος.

Διαφορετικές επιλογές σε κάθε ένα από τα παραπάνω βήματα συνεπάγονται και διαφορετικές μεθόδους για την επίλυση ΠΣΤ. Στην παρούσα εργασία στο προς μελέτη πρόβλημα χαρακτηρίζεται από τις ακόλουθες συγκεκριμένες επιλογές :



Εικόνα 10 : Γεωμετρική απεικόνιση της διαμέρισης του πεδίου Ω

- **Βήμα 0:** Θεωρούμε ομοιόμορφο διαμερισμό των διαστημάτων $I^x = I^y = [0,1]$ σε n_s υποδιαστήματα $I^x_m = I^y_m$, $m = 1$, ... , n_s τα οποία παράγουν ένα ομοιόμορφο πλέγμα με βήμα διακριτοποίησης $h = \frac{1}{n_s}$ και συντεταγμένες κόμβων (x_i, y_j), όπου $x_i = (i-1)h$ και $y_j = (j-1)h$, με $i, j = 1$, ... , $(n_s + 1)$. Η Εικόνα 10 εμφανίζει την διαμέριση του Ω για $n_s = 6$.

- **Βήμα 1:** Ως συναρτήσεις βάσεις επιλέγονται τα Hermite Bicubic πολυώνυμα [20], με γενική μορφή $\Phi_k(x,y) = \Phi_i(x)\Phi_j(y)$, και η συνάρτηση $u(x,y)$ προσεγγίζεται από την

$$u(x,y) \simeq u_n(x,y) = \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{\tilde{n}} a_{i,j} \Phi_i(x) \Phi_j(y) \tag{6}$$

όπου $\tilde{n} = 2(n_s + 1)$.

- **Βήμα 2:** Ως μέθοδος διακριτοποίησης επιλέγεται η μέθοδος της **Collocation** [5, 14], η οποία κατασκευάζει το γραμμικό σύστημα $C\vec{\mathbf{a}} = \vec{\mathbf{b}}$ απαιτώντας οι συνθήκες $\mathcal{L}u_n - f = 0$ και $\mathcal{B}u_n - g = 0$ να ισχύουν για n καθορισμένα εσωτερικά και συνοριακά collocation σημεία.

- **Βήμα 3:** Για την επίλυση του γραμμικού συστήματος $C\vec{\mathbf{a}} = \vec{\mathbf{b}}$ επιλέγεται η επαναληπτική μέθοδος **AOR**.

Στις παραγράφους που ακολουθούν παρουσιάζεται περισσότερο αναλυτικά η τεχνική περιγραφή των παραπάνω επιλογών.

2.2 Πολυώνυμα Hermite

Τα τμηματικά κυβικά πολυώνυμα Hermite ορίζονται ως εξής :

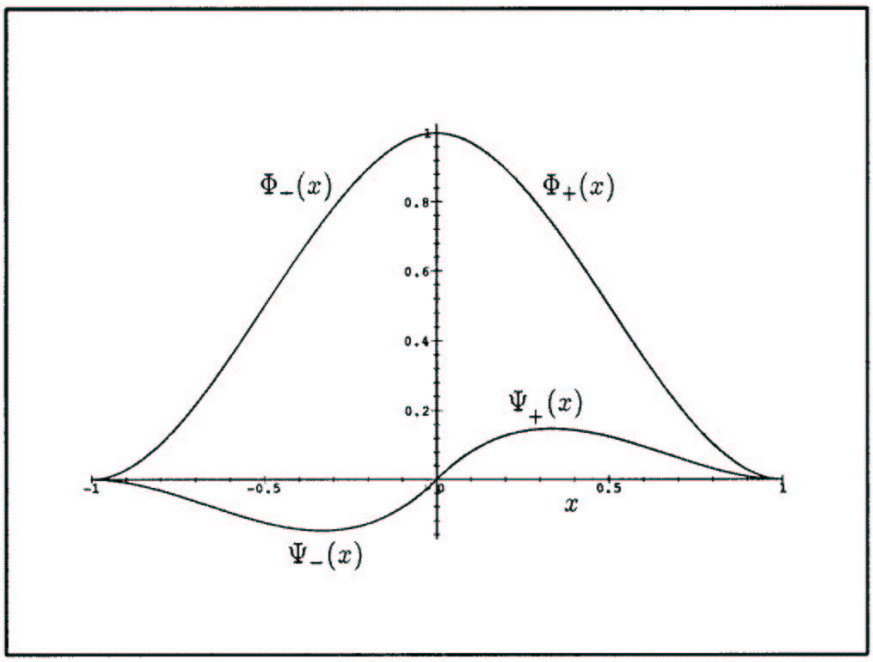
$$\Phi(x) \doteq \begin{cases} \Phi_+(x) & , \quad x \in [0,1] \\ \Phi_-(x) & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,1] \end{cases} , \tag{7}$$

$$\Psi(x) \doteq \begin{cases} \Psi_+(x) & , \quad x \in [0,1] \\ \Psi_-(x) & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,1] \end{cases} , \tag{8}$$

όπου

$$\Phi_+(x) \doteq \begin{cases} (1-x)^2(1+2x) & , \quad x \in [0,1] \\ 0 & , \quad x \notin [0,1] \end{cases} , \tag{9}$$

$$\Phi_-(x) = \Phi_+(-x) \doteq \begin{cases} (1+x)^2(1-2x) & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,0] \end{cases} , \tag{10}$$



Εικόνα 11 : Κυβικά πολυώνυμα Hermite

$$\Psi_+(x) \doteq \begin{cases} x(1-x)^2 & , \quad x \in [0,1] \\ 0 & , \quad x \notin [0,1] \end{cases} , \tag{11}$$

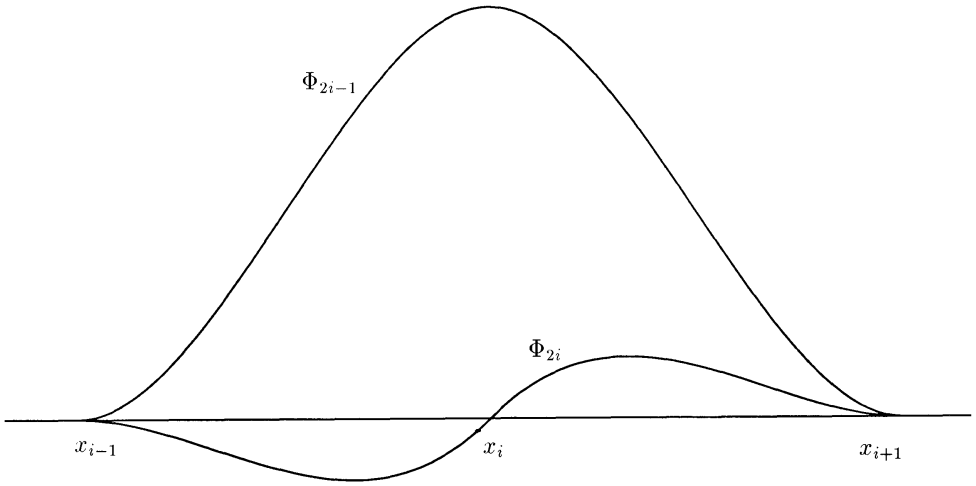
$$\Psi_-(x) = -\Psi_+(-x) \doteq \begin{cases} x(1+x)^2 & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,0] \end{cases} . \tag{12}$$

Η Εικόνα 11 εμφανίζει τα κυβικά πολυώνυμα Hermite, όπως αυτά ορίζονται στο $[-1,1]$.

Σε κάθε κόμβο x_m αντιστοιχούν δύο συναρτήσεις και ορίζονται ως εξής:

$$\Phi_{2m-1}(x) \doteq \begin{cases} \Phi\left(\frac{x_m - x}{h}\right) & , \quad x \in I^x_{m-1} \cup I^x_m \\ 0 & , \quad \text{διαφορετικά} \end{cases} , \tag{13}$$

$$\Phi_{2m}(x) \doteq \begin{cases} \Psi\left(\frac{x - x_m}{h}\right) & , \quad x \in I^x_m \cup I^x_{m-1} \\ 0 & , \quad \text{διαφορετικά} \end{cases} , \tag{14}$$



Εικόνα 12 : Πολυώνυμα Hermite ορισμένα στον κόμβο x_i.

όπου $m = 1, \ldots, (n_s + 1)$, $I_i^x \equiv [x_i, x_{i+1}]$, $i = 1, \ldots, n_s$.

Για να ισχύουν οι ορισμοί (13) και (14) για $m = 1$ και $m = n_s + 1$, θεωρούμε δύο υποθετικούς κόμβους $x_0 = -h$ και $x_{n_s+2} := 1 + h$.

Στην Εικόνα 12 εμφανίζεται ένας τυχαίος κόμβος x_i και παρουσιάζονται οι αντίστοιχες συναρτήσεις όπως αυτές ορίζονται στο κόμβο αυτόν.

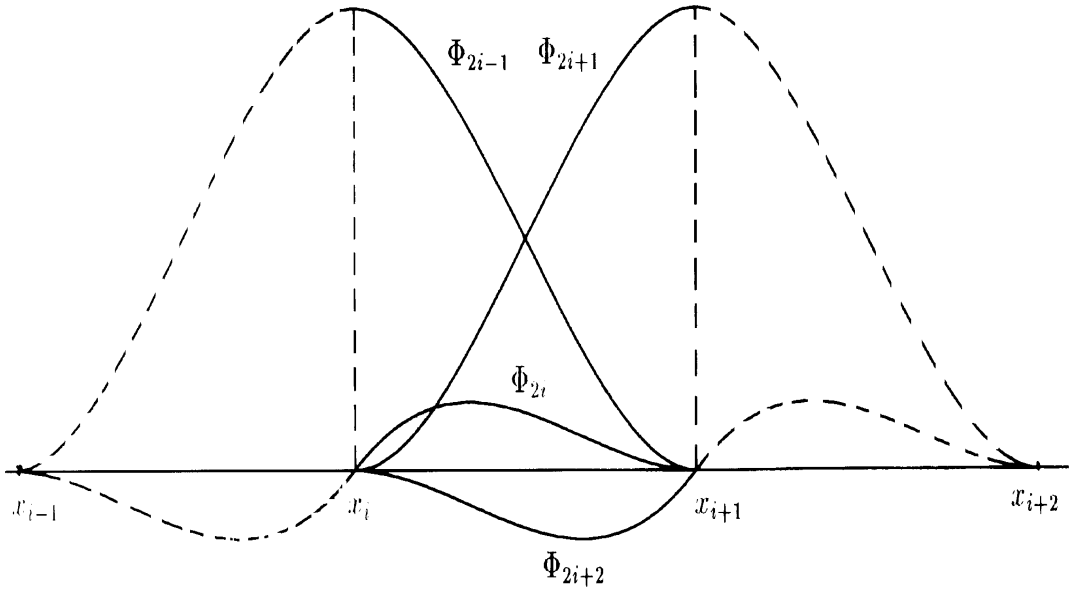
Παρατηρούμε ότι ισχύουν οι εξής ιδιότητες

$$\begin{cases} \Phi_{2m-1}(x_i) &= h\frac{d}{dt}\Phi_{2m}(x_i) = \delta_m^i \\ \Phi_{2m}(x_i) &= \frac{d}{dt}\Phi_{2m-1}(x_i) = 0 \end{cases} \tag{15}$$

για όλα τα $m = 1, \ldots, (n_s + 1)$ και όπου δ_m^i : Δέλτα του Kronecker.

Επιπλέον σημειώνουμε ότι σε κάθε υποδιάστημα I_i^x διέρχονται τέσσερα μόνο μη μηδενικά πολυώνυμα Hermite με δείκτες : Φ_{2i-1} , Φ_{2i} , Φ_{2i+1} και Φ_{2i+2}.

Το γεγονός αυτό επιδεικνύεται σχηματικά και στην Εικόνα 13.



Εικόνα 13 : Μη μηδενικά Πολυώνυμα Hermite στο υποδιάστημα $[x_i, x_{i+1}]$.

Βασισμένοι στις παραπάνω ιδιότητες των πολυωνύμων Hermite εύκολα προκύπτουν και οι ιδιότητες των διδιάστατων bicubic πολυωνύμων Hermite. Έτσι, παρατηρούμε ότι σε κάθε διδιάστατο κόμβο (x_i, y_j) ορίζονται τα παρακάτω τέσσερα Hermite Bicubic πολυώνυμα :

$$\begin{cases} \Phi_{2i-1,2j-1}(x,y) &= \Phi_{2i-1}(x)\Phi_{2j-1}(y) \\ \Phi_{2i-1,2j}(x,y) &= \Phi_{2i-1}(x)\Phi_{2j}(y) \\ \Phi_{2i,2j-1}(x,y) &= \Phi_{2i}(x)\Phi_{2j-1}(y) \\ \Phi_{2i,2j}(x,y) &= \Phi_{2i}(x)\Phi_{2j}(y) \end{cases} \tag{16}$$

με τις εξής ιδιότητες :

$$\begin{cases} \Phi_{2i-1,2j-1}(x_i,y_j) = 1 \quad , \quad h\frac{\partial}{\partial y}\Phi_{2i-1,2j}(x_i,y_j) = 1 \\ h\frac{\partial}{\partial x}\Phi_{2i,2j-1}(x_i,y_j) = 1 \quad , \quad h^2\frac{\partial^2}{\partial x \partial y}\Phi_{2i,2j}(x_i,y_j) = 1 \end{cases} \tag{17}$$

Σαν άμεση συνέπεια των προηγουμένων σχέσεων προκύπτει ότι

$$\begin{cases} u_n(x_i,y_j) & = & a_{2i-1,2j-1} & , & h\frac{\partial}{\partial y}u_n(x_i,y_j) & = & a_{2i-1,2j} \\ h\frac{\partial}{\partial x}u_n(x_i,y_j) & = & a_{2i,2j-1} & , & h^2\frac{\partial^2}{\partial x\partial y}u_n(x_i,y_j) & = & a_{2i,2j} \end{cases} \tag{18}$$

όπου $i\ ,\ j\ =\ 1,\ldots,(n_s+1)$.

Επιπλέον παρατηρούμε ότι σε κάθε στοιχείο I_{ij}^{xy} αντιστοιχούν 16 μη μηδενικές συναρτήσεις βάσης και επομένως για $(x,y)\in I_{ij}^{xy}$ ισχύει ότι :

$$u_n(x,y)\ =\ \sum_{k=2i-1}^{2i+2}\sum_{l=2j-1}^{2j+2}\alpha_{k,l}\Phi_k(x)\Phi_l(y)\ . \tag{19}$$

Γι' αυτό το λόγο το στοιχείο I_{ij}^{xy} είναι στοιχείο με 16 βαθμούς ελευθερίας.

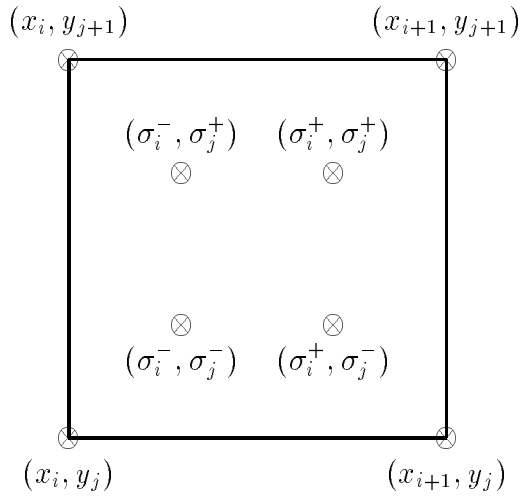
2.3 Η μέθοδος Collocation

Οι Collocation εξισώσεις κατασκευάζονται απαιτώντας το υπόλοιπο $\mathcal{L}u_n - f$ να μηδενίζεται σε $n_I = 4n_s^2$ **εσωτερικά** collocation σημεία και το υπόλοιπο $\mathcal{B}u_n - g$ να μηδενίζεται σε $n_b = 4(2n_s+1)$ **συνοριακά** collocation σημεία. Παρατηρούμε ότι το πλήθος $n_I + n_b$ των collocation εξισώσεων ισούται με τον αριθμό των αγνώστων, ο οποίος προκύπτει από τη χρήση των πολυωνύμων Hermite Bicubic.

Πρέπει να τονίσομε εδώ, ότι στην περίπτωση των Dirichlet συνοριακών συνθηκών κάποιοι από τους αγνώστους, οι οποίοι βρίσκονται πάνω στο σύνορο του Ω προσδιορίζονται άμεσα από αυτές, με αποτέλεσμα την απαλοιφή τους από το γραμμικό σύστημα. Το πλήθος των αγνώστων αυτών είναι $4(2n_s+1)$ με αποτέλεσμα να μην χρειάζεται η επιλογή συνοριακών collocation σημείων.

Οπότε για την Dirichlet περίπτωση το πλήθος των αγνώστων και συνεπώς και των εσωτερικών collocation σημείων ισούται με $n = 4n_s^2$.

Στην περίπτωση ελλειπτικών ΠΣΤ η standard επιλογή εσωτερικών collocation σημείων είναι αυτή των σημείων Gauss [5] με συντεταγμένες $(\sigma_i^{\pm},\sigma_j^{\pm})$, όπου για κάθε $i=1,\ldots,n_s$ ισχύει ότι $\sigma_i^{\pm} = 2i-1\pm\frac{\sqrt{3}}{3}$. Στην Εικόνα 14 εμφανίζονται τα τέσσερα σημεία Gauss του στοιχείου I_{ij}^{xy}.



Εικόνα 14 : Τα τέσσερα σημεία Gauss στο πεπερασμένο στοιχείο I_{ij}^{xy}

Η αρίθμηση αγνώστων και εξισώσεων καθορίζει την δομή του collocation πίνακα, και κατά συνέπεια την επιλογή της μεθόδου επίλυσης του παραγόμενου γραμμικού συστήματος. Η Εικόνα 15 παρουσιάζει την αρίθμηση αγνώστων και η Εικόνα 16 την αρίθμηση των εξισώσεων σύμφωνα με την standard collocation μέθοδο. Οι άγνωστοι, οι οποίοι έχουν απαλοιφθεί εξαιτίας των συνοριακών συνθηκών σημειώνονται με "x" .

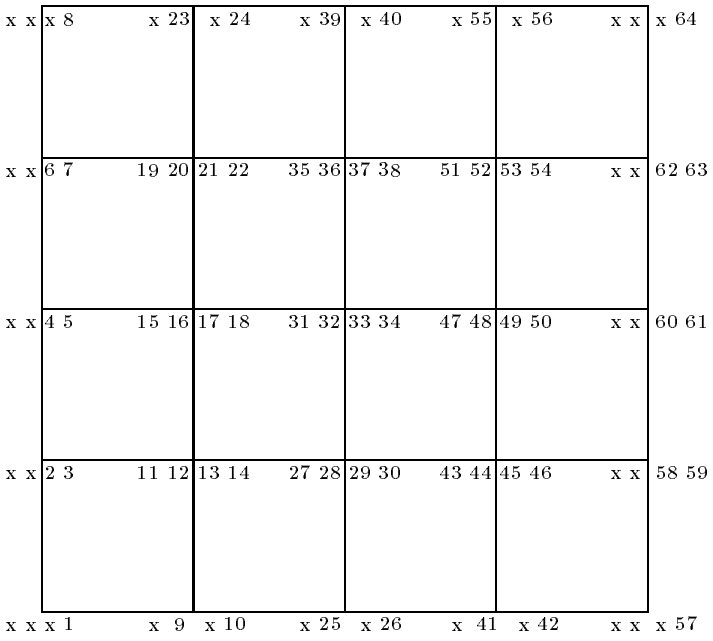
Η δομή του standard collocation πίνακα εμφανίζεται στην Εικόνα 17. Εύκολα συνάγεται ότι η δομή αυτή δεν είναι κατάλληλη για την χρήση επαναληπτικών μεθόδων. Στην εργασία [21] ο Θ. Σ. Παπαθεοδώρου πρότεινε την αρίθμηση των αγνώστων, η οποία παρουσιάζεται στην Εικόνα 19, και των εξισώσεων, η οποία παρουσιάζεται στην Εικόνα 20, ώστε να προκύψει ο collocation block τριδιαγώνιος πίνακας του οποίου η δομή εμφανίζεται στην Εικόνα 18.

Κάνοντας χρήση της μεθόδου αρίθμησης του Παπαθεοδώρου στο Dirichlet - Poisson πρόβλημα παράγεται το ακόλουθο γραμμικό σύστημα

$$A\boldsymbol{x} \;=\; \boldsymbol{b} \;, \tag{20}$$

όπου $A \in \mathbb{R}^{n,n}$ $(n = 4{n_s}^2)$ είναι ο Collocation πίνακας συντελεστών και

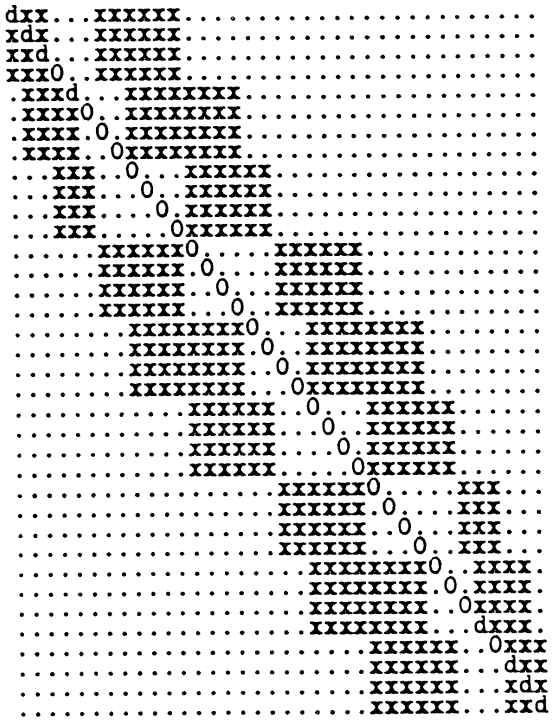
$\boldsymbol{x} \;=\; [x_1 \;\; x_2 \;\; \cdots \;\; x_n]^T \;=\; [\alpha_{1,1} \;\; \cdots \;\; \alpha_{2n_s,2n_s}]^T$ είναι το διάνυσμα των αγνώστων.



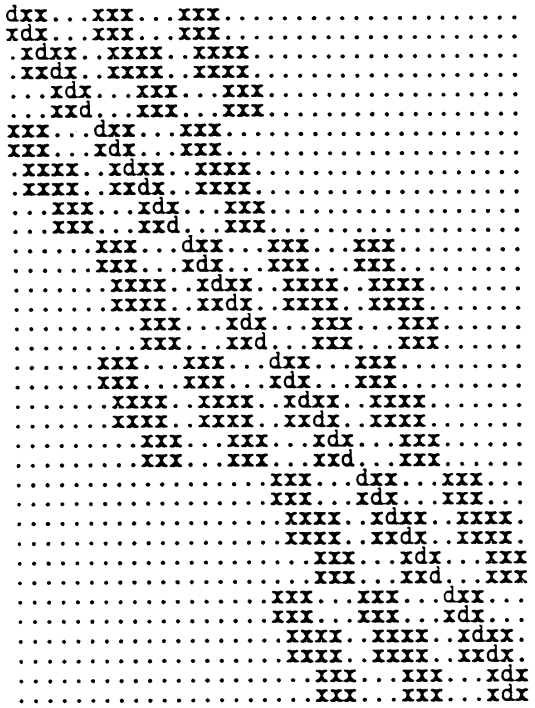
Εικόνα 15 : Standard Αρίθμηση αγνώστων για $n_s = 4$

14	16	30	32	46	48	62	64
13	15	29	31	45	47	61	63
10	12	26	28	42	44	58	60
9	11	25	27	41	43	57	59
6	8	22	24	38	40	54	56
5	7	21	23	37	39	53	55
2	4	18	20	34	36	50	52
1	3	17	19	33	35	49	51

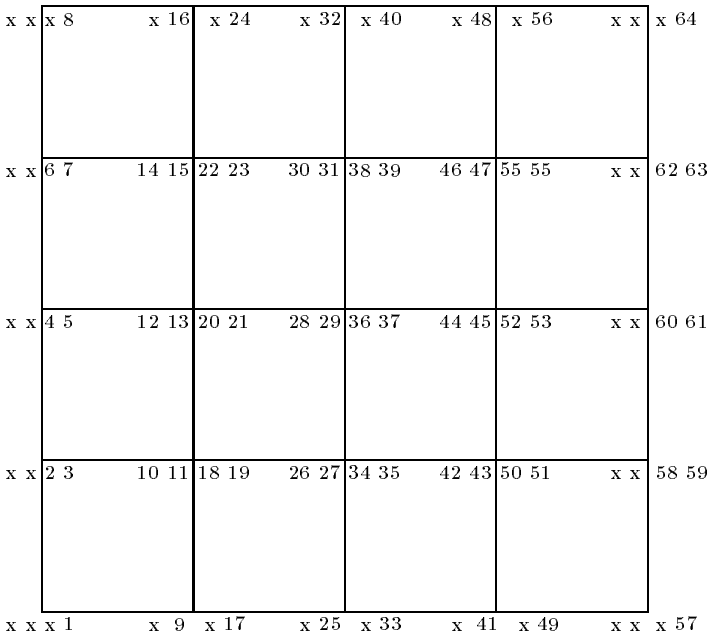
Εικόνα 16 : Standard Αρίθμηση εξισώσεων για $n_s = 4$



Εικόνα 17 : Δομή του Standard Collocation Πίνακα για $n_s = 3$



Εικόνα 18 : Δομή του Block Τριδιαγώνιου Collocation Πίνακα για $n_s = 3$



Εικόνα 19 : Block Τριδιαγώνια Αρίθμηση αγνώστων για $n_s = 4$

8	16	24	32	40	48	56	64
7	15	23	31	39	47	55	63
6	14	22	30	38	46	54	62
5	13	21	29	37	45	53	61
4	12	20	28	36	44	52	60
3	11	19	27	35	43	51	59
2	10	18	26	34	42	50	58
1	9	17	25	33	41	49	57

Εικόνα 20 : Block Τριδιαγώνια Αρίθμηση εξισώσεων για $n_s = 4$

Πιο συγκεκριμένα, θεωρώντας ότι έχει βγει κοινός παράγοντας το $1/9h^2$, αρχικά ορίζομε τους $2n_s \times 2n_s$ $seed$ πίνακες

$$A_i = \begin{bmatrix} a_2 & a_3 & -a_4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ a_4 & a_1 & -a_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & -a_4 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & a_4 & a_1 & -a_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_1 & a_2 & a_3 & -a_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_3 & a_4 & a_1 & -a_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_1 & a_2 & -a_4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_3 & a_4 & -a_2 \end{bmatrix} \quad , \quad i = 1, 2, 3, 4 \quad (21)$$

όπου οι τιμές a_i's δίνονται ως[19, 21]

Table 1	a_1	a_2	a_3	a_4
A_1	$-r^+$	$-s^+$	q	t^+
A_2	$-s^+$	$-u^+$	t^-	0
A_3	q	t^-	$-r^-$	$-s^-$
A_4	t^+	0	$-s^-$	$-u^-$

(22)

με $q = 24$, $r^{\pm} = 24 \pm 18\sqrt{3}$, $s^{\pm} = 12 \pm 8\sqrt{3}$, $t^{\pm} = 3 \pm \sqrt{3}$, $u^{\pm} = 3 \pm 2\sqrt{3}$.

Κάνοντας χρήση των παραπάνω ορισμών ο block τριδιαγώνιος collocation πίνακας ορίζεται ως εξής

$$A = \begin{bmatrix} A_2 & A_3 & -A_4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ A_4 & A_1 & -A_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & A_1 & A_2 & A_3 & -A_4 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & A_3 & A_4 & A_1 & -A_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & A_1 & A_2 & A_3 & -A_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & A_3 & A_4 & A_1 & -A_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & A_1 & A_2 & -A_4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & A_3 & A_4 & -A_2 \end{bmatrix} . \quad (23)$$

Είναι γνωστό ότι για block τριδιαγώνιους πίνακες μπορεί κανείς να βρει πλούσια βιβλιογραφία [10,15-17,21,23-25] για την εφαρμογή και ανάλυση επαναληπτικών μεθόδων.

2.4 Επαναληπτική μέθοδος AOR

Για την επίλυση του γραμμικού συστήματος

$$A\boldsymbol{x} \;=\; \boldsymbol{b} \tag{24}$$

όπου ο πίνακας $A \in \mathbb{R}^{n,n}$ είναι αντιστρέψιμος και $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$, αρχικά θεωρούμε την παρακάτω διάσπασή του

$$\mathrm{A} = D_A - L_A - U_A \tag{25}$$

όπου ο πίνακας D_A είναι κάποιο αντιστρέψιμο block διαγώνιο τμήμα του πίνακα A αντίστοιχα.
Βασιζόμενοι στη παρακάτω διάσπαση του πίνακα A

$$\mathrm{A} = \frac{1}{\omega}(D_A - rL_A) - \frac{1}{\omega}[(1-\omega)D_A + (\omega - r)L_A + \omega U_A] \tag{26}$$

όπου $\omega \neq 0$, η επαναληπτική μέθοδος AOR ορίζεται ως εξής

$$\begin{cases} \boldsymbol{x}^{(m+1)} &= \mathcal{L}_{r,\omega}\boldsymbol{x}^{(m)} \;+\; \boldsymbol{c}_{r,\omega} \;\;,\; m = 0, 1, \ldots \\ \mathcal{L}_{r,\omega} &= (D_A - rL_A)^{-1}[(1-\omega)D_A + (\omega - r)L_A + \omega U_A] \quad , \\ \boldsymbol{c}_{r,\omega} &= \omega(D_A - rL_A)^{-1}\boldsymbol{b} \end{cases} \tag{27}$$

όπου $\boldsymbol{x}^{(0)}$ είναι κάποια αρχική προσέγγιση της λύσης, $\mathcal{L}_{r,\omega}$ είναι ο AOR επαναληπτικός πίνακας και οι παράμετροι r, ω αναφέρονται ως $acceleration$ και $overrelaxation$ παράγοντες αντίστοιχα. Για την περίπτωση κατά την οποία $\omega = r$ προκύπτει η επαναληπτική μέθοδος SOR.

Η διάσπαση του πίνακα A σύμφωνα με την (26) εγγυάται ότι η μέθοδος AOR είναι completely consistent. Η συνθήκη σύγκλισης $\rho(\mathcal{L}_{r,\omega}) < 1$ καθώς και η ταχύτητα σύγκλισης της μεθόδου AOR δεν εξαρτάται μόνο από την επιλογή των παραμέτρων r και ω, αλλά και από την επιλογή των πινάκων διάσπασης D_A, L_A και U_A, δηλαδή από τη **διαμέριση** του πίνακα A.

Για την παρακάτω block διαμέριση του τριδιαγώνιου collocation πίνακα A,

$$\left[\begin{array}{cc|cc|ccc|cc|cc}
A_2 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\
A_4 & A_1 & -A_2 & O & O & \cdots & O & O & O & O & O \\
\hline
O & A_1 & A_2 & A_3 & -A_4 & \cdots & O & O & O & O & O \\
O & A_3 & A_4 & A_1 & -A_2 & \cdots & O & O & O & O & O \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
O & O & O & O & O & \cdots & A_1 & A_2 & A_3 & -A_4 & O \\
O & O & O & O & O & \cdots & A_3 & A_4 & A_1 & -A_2 & O \\
\hline
O & O & O & O & O & \cdots & O & O & A_1 & A_2 & -A_4 \\
O & O & O & O & O & \cdots & O & O & A_3 & A_4 & -A_2
\end{array}\right]$$

η οποία προτάθηκε από τον Θ. Σ. Παπαθεοδώρου [21], οι πίνακες διάσπασης θα είναι

$$D_A = diag[D \quad D \quad \ldots \quad D \quad \hat{D}] \quad , \tag{28}$$

$$L_A = -\begin{bmatrix}
O & O & O & \ldots & O & O & O \\
L & O & O & \ldots & O & O & O \\
O & L & O & \ldots & O & O & O \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & \ldots & L & O & O \\
O & O & O & \ldots & O & L & O
\end{bmatrix} \quad , \tag{29}$$

$$U_A = -\begin{bmatrix}
O & U & O & \ldots & O & O & O \\
O & O & U & \ldots & O & O & O \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & \ldots & O & U & O \\
O & O & O & \ldots & O & O & U \\
O & O & O & \ldots & O & O & O
\end{bmatrix} \tag{30}$$

όπου

$$D = \begin{bmatrix} A_2 & A_3 \\ A_4 & A_1 \end{bmatrix} \quad , \quad \hat{D} = \begin{bmatrix} A_2 & -A_4 \\ A_4 & -A_2 \end{bmatrix} \quad ,$$

$$L = \begin{bmatrix} O & A_1 \\ O & A_3 \end{bmatrix} \quad , \quad U = \begin{bmatrix} -A_4 & O \\ -A_2 & O \end{bmatrix} \quad .$$

Η ανάλυση της σύγκλισης της μεθόδου AOR και ο καθορισμός των βέλτιστων τιμών των παραμέτρων παρουσιάζονται στις [21].

Όπως έχομε προαναφέρει η διαμέριση του πίνακα A αποτελεί σημαντικό παράγοντα στην ταχύτητα σύγκλισης της μεθόδου. Έτσι στην πρόσφατη εργασία των Lai, Hadjidimos, Houstis και Rice [15] διερευνήθηκε η αποτελεσματικότητα διαφορετικών διαμερίσεων του πίνακα A και απεδείχθει ότι με τη διαμέριση

$$\left[\begin{array}{c|ccc|cc|cc|c} A_2 & A_3 & -A_4 & O & O & \cdots & O & O & O & O \\ \hline A_4 & A_1 & -A_2 & O & O & \cdots & O & O & O & O \\ O & A_1 & A_2 & A_3 & -A_4 & \cdots & O & O & O & O \\ \hline O & A_3 & A_4 & A_1 & -A_2 & \cdots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & A_1 & A_2 & A_3 & -A_4 \\ \hline O & O & O & O & O & \cdots & A_3 & A_4 & A_1 & -A_2 \\ O & O & O & O & O & \cdots & O & O & A_1 & A_2 & -A_4 \\ \hline O & O & O & O & O & \cdots & O & O & A_3 & A_4 & -A_2 \end{array}\right]$$

επιτυγχάνεται πολύ μεγαλύτερη ταχύτητα σύγκλισης. Με βάση την παραπάνω διαμέριση οι πίνακες διάσπασης D_A, L_A και U_A ορίζονται από τις σχέσεις

$$D_A \;=\; diag[A_2 \quad Q \quad \ldots \quad Q \quad -A_2] \quad , \tag{31}$$

όπου $Q \;=\; \begin{bmatrix} A_1 & -A_2 \\ A_1 & A_2 \end{bmatrix}$ και

$$L_A \;=\; -\begin{bmatrix} O & O & O & \ldots & O & O & O & O & O \\ A_4 & O & O & \ldots & O & O & O & O & O \\ O & O & O & \ldots & O & O & O & O & O \\ O & A_3 & A_4 & \ldots & O & O & O & O & O \\ O & O & O & \ldots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & \ldots & A_3 & A_4 & O & O & O \\ O & O & O & \ldots & O & O & O & O & O \\ O & O & O & \ldots & O & O & A_3 & A_4 & O \end{bmatrix} \quad , \tag{32}$$

$$U_A \;=\; -\begin{bmatrix} O & A_3 & -A_4 & O & O & \ldots & O & O & O \\ O & O & O & O & O & \ldots & O & O & O \\ O & O & O & A_3 & -A_4 & \ldots & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \ldots & O & O & O \\ O & O & O & O & O & \ldots & A_3 & -A_4 & O \\ O & O & O & O & O & \ldots & O & O & O \\ O & O & O & O & O & \ldots & O & O & -A_4 \\ O & O & O & O & O & \ldots & O & O & O \end{bmatrix} \tag{33}$$

Η παραπάνω διαμέριση θα χρησιμοποιηθεί στις επόμενες παραγράφους για την **παράλληλη** εφαρμογή της επαναληπτικής μεθόδου AOR και την επίλυση του collocation συστήματος.

3 Παράλληλοι Αλγόριθμοι και εφαρμογές

3.1 Εισαγωγή

Από την εποχή ακόμη των πρώτων παράλληλων μηχανών ένα από τα κύρια θέματα έρευνας απετέλεσε το πέρασμα από το σειριακό πρόβλημα στο παράλληλο [3]. Και τούτο διότι οι σειριακοί αλγόριθμοι αδυνατούν να εκμεταλλευτούν τα πλεονεκτήματα που προσφέρουν οι παράλληλες μηχανές [18]. Για να γίνουμε πιο συγκεκριμένοι ας εξετάσομε την δική μας σειριακή εφαρμογή και τα προβλήματα που δημιουργεί η απεικόνησή της σε παράλληλα περιβάλλοντα.

Από τον ορισμό της μεθόδου AOR της σχέσης (27) έπεται ότι σε κάθε επαναληπτικό βήμα έχουμε να επιλύσουμε ένα γραμμικό σύστημα της μορφής

$$(D_A - rL_A)\boldsymbol{x}^{(m+1)} = [(1-\omega)D_A + (\omega - r)L_A + \omega U_A]\boldsymbol{x}^{(m)} + \omega \boldsymbol{b} \qquad (34)$$

Ανακαλώντας τώρα τους ορισμούς των πινάκων D_A και L_A από τις σχέσεις (31-32) αντίστοιχα, είναι σαφές ότι ο πίνακας $D_A - rL_A$ είναι block κάτω τριγωνικός και επομένως σε κάθε βήμα είναι αναγκαία η επίλυση ενός block τριγωνικού συστήματος. Αυτό έχει σαν αποτέλεσμα ότι για τον υπολογισμό του

$\boldsymbol{x}_k^{(m+1)}$ προαπαιτείται ο υπολογισμός όλων των προηγούμενων αγνώστων $\boldsymbol{x}_1^{(m+1)}$, $\boldsymbol{x}_2^{(m+1)} \ldots \boldsymbol{x}_{k-1}^{(m+1)}$ γεγονός που αποκλείει τον παράλληλο υπολογισμό του $\boldsymbol{x}_k^{(m+1)}$ με κάποιο άλλο υποδιάνυσμα αγνώστων $\boldsymbol{x}_j^{(m+1)}$.

Γίνεται προφανές ότι το υπολογιστικό μας πρόβλημα στην σειριακή του μορφή, δεν είναι εφαρμόσιμο σε παράλληλα περιβάλλοντα. Το κύριο θέμα της ενότητας αυτής είναι η προσπάθεια παραλληλοποίησης της εφαρμογής μας και η απεικόνηση της σε μια παράλληλη μηχανή μέσω του PVM και σε ένα Distributed Memory MIMD παράλληλο υπολογιστικό σύστημα.

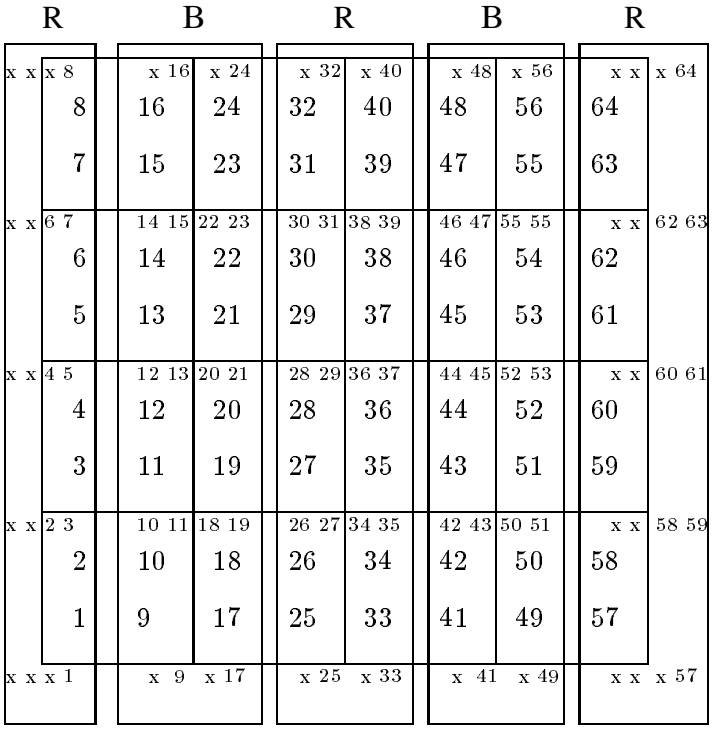
3.2 Παραλληλοποίηση της εφαρμογής

Η διαδικασία παραλληλοποίησης της εφαρμογής μας χωρίζεται σε δυο κύριες φάσεις. Στην πρώτη φάση χρησιμοποιούμε με επιτυχία την ιδέα **επαναρίθμησης των αγνώστων και εξισώσεων** για να παραλληλοποιήσομε σ' ένα πρώτο βαθμό το υπολογιστικό πρόβλημα. Στη συνέχεια κάνομε χρήση της έννοιας του **preconditioning** για να βελτιστοποιήσομε το βαθμό παραλληλοποίησης της εφαρμογής.

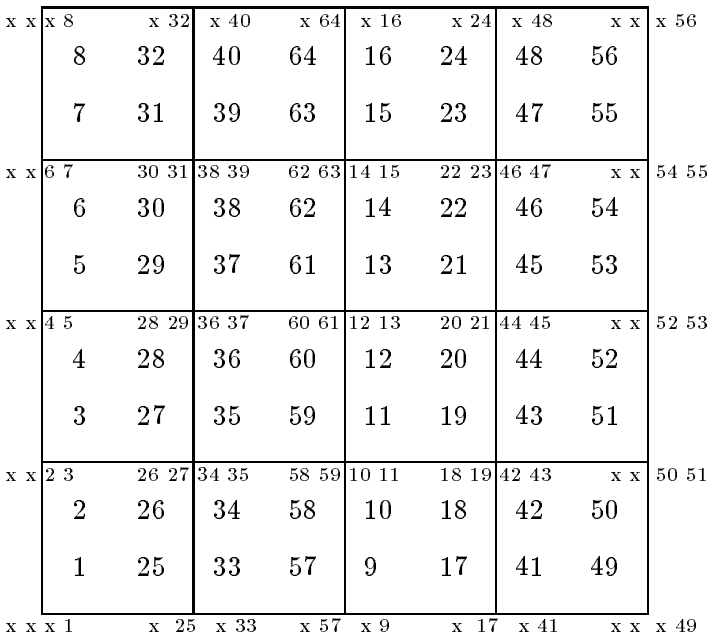
Για την επαναρίθμηση των αγνώστων και των εξισώσεων χρησιμοποιούμε [11, 25] την γνωστή ιδέα της line red - black διαμέρισης. Σύμφωνα με αυτήν οι άγνωστοι / εξισώσεις χωρίζονται σε red και black υποομάδες αγνώστων / εξισώσεων με την παρακάτω αρχή :

Τα μέλη red υποομάδων "συνορεύουν" μόνο με μέλη black υποομάδων

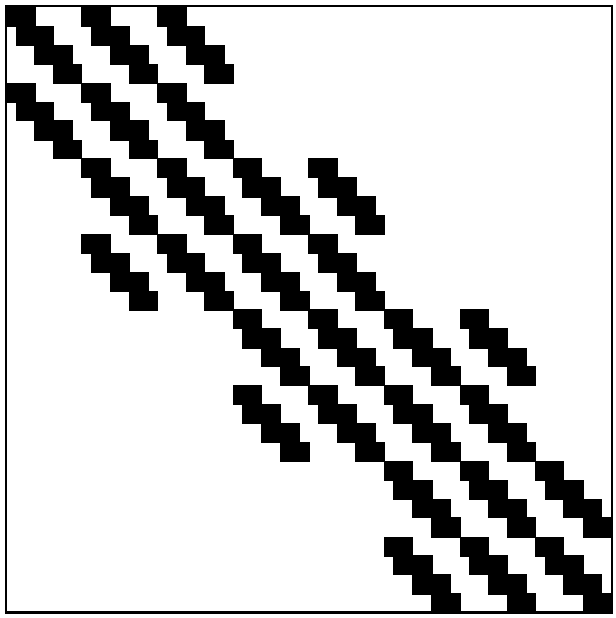
Η Εικόνα 21 εμφανίζει την ομαδοποίηση αυτή για την περίπτωση $n_s = 4$. Στην συνέχεια επαναριθμούμε τους αγνώστους και τις εξισώσεις, έτσι ώστε τα μέλη των red υποομάδων να καταλάβουν διαδοχικές θέσεις στο grid, ακολουθούμενα με την ίδια ιδέα από τα μέλη των black υποομάδων. Στην Εικόνα 22 παρουσιάζεται η νέα αυτή αρίθμηση αγνώστων και εξισώσεων για $n_s = 4$. Η δομή των πινάκων που αντιστοιχούν στην αρίθμηση των Εικόνων 21 και 22 παρουσιάζεται αντίστοιχα στις Εικόνες 23 και 24.



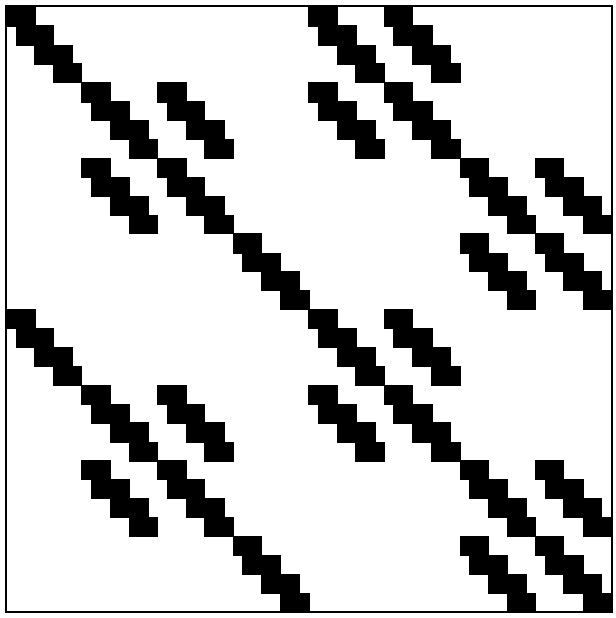
Εικόνα 21 : Red - Black ομαδοποίηση αγνώστων και εξισώσεων



Εικόνα 22 : Red - Black αρίθμηση αγνώστων και εξισώσεων



Εικόνα 23 : Δομή του Block Τριδιαγώνιου Collocation Πίνακα



Εικόνα 24 : Δομή του Red - Black Collocation Πίνακα

Έτσι ο πίνακας A της σχέσης (23), ο οποίος αντιστοιχεί στη block τριδιαγώνια αρίθμηση της Εικόνας 21, μετασχηματίζεται στον παρακάτω πίνακα

$$A = \begin{bmatrix} A_2 & O & O & \dots & O & O & O & A_3 & -A_4 & \dots & O & O \\ O & A_1 & -A_2 & \dots & O & O & O & A_3 & A_4 & \dots & O & O \\ O & A_1 & A_2 & \dots & O & O & O & O & O & \dots & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & O & \dots & A_1 & -A_2 & O & O & O & \dots & O & O \\ O & O & O & \dots & A_1 & A_2 & O & O & O & \dots & A_3 & -A_4 \\ O & O & O & \dots & O & O & -A_2 & O & O & \dots & A_3 & A_4 \\ A_4 & O & O & \dots & O & O & O & A_1 & -A_2 & \dots & O & O \\ O & A_3 & -A_4 & \dots & O & O & O & A_1 & A_2 & \dots & O & O \\ O & A_3 & A_4 & \dots & O & O & O & O & O & \dots & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & O & \dots & A_3 & -A_4 & O & O & O & \dots & O & O \\ O & O & O & \dots & A_3 & A_4 & O & O & O & \dots & A_1 & -A_2 \\ O & O & O & \dots & O & O & -A_4 & O & O & \dots & A_1 & A_2 \end{bmatrix} \quad ,(35)$$

ο οποίος ισοδύναμα γράφεται σαν

$$A = \begin{bmatrix} D_1 & -H_2 \\ -H_1 & D_2 \end{bmatrix} \quad , \tag{36}$$

όπου

$$H_1 = -\text{diag}[A_4 \; \hat{A} \; \cdots \; \hat{A} \; - A_4 \,] \quad , \tag{37}$$

$$H_2 = -\text{diag}[\hat{A} \; \cdots \; \hat{A}] \quad , \tag{38}$$

με $\hat{A} = \begin{bmatrix} A_3 & -A_4 \\ A_3 & A_4 \end{bmatrix}$

και οι τετραγωνικοί block διαγώνιοι και αντιστέψιμοι [20] πίνακες D_1 και D_2 ορίζονται ως εξής

$$D_1 = \text{diag}[A_2 \; \hat{B} \; \cdots \; \hat{B} \; - A_2 \,] \quad , \tag{39}$$

$$D_2 = \text{diag}[\hat{B} \; \cdots \; \hat{B}] \quad , \tag{40}$$

όπου $\hat{B} = \begin{bmatrix} A_1 & -A_2 \\ A_1 & A_2 \end{bmatrix}$.

Στο σημείο αυτό θα πρέπει να τονίσουμε ότι η επαναρίθμηση αγνώστων και εξισώσεων ισοδυναμεί με έναν μετασχηματισμό ομοιότητας του πίνακα A. Ο μετασχηματισμός αυτός είναι ίδιος με το μετασχηματισμό που μετατρέπει έναν block τριδιαγώνιο πίνακα στην 2-κυκλική κανονική μορφή του [23, 24]. Δηλαδή

$$A \leftarrow PAP^{-1} \tag{41}$$

όπου P είναι ο ακόλουθος μεταθετικός πίνακας

$$P = \begin{bmatrix} I_1 & O & O & O & O & O & \dots & O & O & O & O \\ O & O & I_3 & O & O & O & \dots & O & O & O & O \\ O & O & O & O & I_5 & O & \dots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & O & \dots & O & O & O & O \\ O & O & O & O & O & O & \dots & O & I_{n_s-1} & O & O \\ O & O & O & O & O & O & \dots & O & O & O & I_{n_s+1} \\ O & I_2 & O & O & O & O & \dots & O & O & O & O \\ O & O & O & I_4 & O & O & \dots & O & O & O & O \\ O & O & O & O & O & I_6 & \dots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & O & \dots & I_{n_s-2} & O & O & O \\ O & O & O & O & O & O & \dots & O & O & I_{n_s} & O \end{bmatrix}, \tag{42}$$

όπου $\mathrm{I}_j \in \mathbb{R}^{\nu,\nu}$ με $\nu = \begin{cases} 2n_s & \text{όταν} \quad j = 1 \text{ και } j = n_s + 1 \\ 4n_s & \text{όταν} \quad j = 2, \dots, j = n_s \end{cases}$ για $j = 1, \dots, n_s + 1$.

Έτσι και τα υπόλοιπα μέλη του γραμμικού συστήματος μετασχηματίζονται ανάλογα :

$$\boldsymbol{x} \leftarrow P\boldsymbol{x} \qquad \text{και} \qquad \boldsymbol{b} \leftarrow P\boldsymbol{b} \tag{43}$$

Εξετάζοντας τώρα λεπτομερέστερα την μορφή των διαγώνιων πινάκων D_1, D_2 και $\hat{B}$ είναι προφανές ότι κανείς μπορεί να χρησιμοποιήσει preconditioning για την παραπέρα απεξάρτηση των αγνώστων με άμεση συνέπεια την αύξηση του βαθμού παραληλοποίησης.

Σ' αυτήν την κατεύθυνση και σύμφωνα με την διαμέριση του πίνακα A από την (31) ορίζουμε τον πίνακα

$$T = \text{diag}\,[\,T_1 \;\; T_2\,] \tag{44}$$

με

$$T_1 \;=\; \text{diag}[I\; G\; \cdots\; G\; -I] \quad , \quad T_2 \;=\; \text{diag}[G\; \cdots\; G] \;\; , \tag{45}$$

όπου

$$G \;=\; \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix} \;\; . \tag{46}$$

Το preconditioned σύστημα γράφεται τώρα σαν

$$TA\boldsymbol{x} \;=\; T\boldsymbol{b} \tag{47}$$

ή, θεωρώντας ότι τα διανύσματα $\boldsymbol{x} = [\boldsymbol{x}_R \quad \boldsymbol{x}_B]^T$ και $\boldsymbol{b} = [\boldsymbol{b}_R \quad \boldsymbol{b}_B]^T$ έχουν διαμεριστεί σύμφωνα με το διαμερισμό του πίνακα A, ισοδύναμα

$$\begin{bmatrix} T_1 & O \\ O & T_2 \end{bmatrix} \begin{bmatrix} D_1 & -H_2 \\ -H_1 & D_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} \;=\; \begin{bmatrix} T_1 & O \\ O & T_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_R \\ \boldsymbol{b}_B \end{bmatrix} \;\; . \tag{48}$$

Δηλαδή

$$\begin{bmatrix} D & -\bar{F} \\ -F & \bar{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} \;=\; \begin{bmatrix} T_1\tilde{\boldsymbol{b}}_R \\ T_2\tilde{\boldsymbol{b}}_B \end{bmatrix} \tag{49}$$

όπου έχουμε θεωρήσει ότι

$$D \;=\; T_1 D_1 \;=\; \text{diag}[A_2\; A_1\; A_2\; \cdots\; A_1\; A_2\;\; A_2\;] \;\; , \tag{50}$$

$$\bar{D} \;=\; T_2 D_2 \;=\; \text{diag}[A_1\; A_2\; \cdots\; A_1\; A_2\;] \;\; , \tag{51}$$

$$\tilde{\boldsymbol{b}}_R \;=\; T_1\boldsymbol{b}_1 \;=\; \frac{1}{2}\begin{bmatrix} 2b_1 \\ b_2 + b_3 \\ b_3 - b_2 \\ \\ \vdots \\ \\ b_{n_s-2} + b_{n_s-1} \\ b_{n_s-1} - b_{n_s-2} \\ -2b_{n_s} \end{bmatrix} \;\; , \tag{52}$$

$$\tilde{\boldsymbol{b}}_B \;=\; T_2\boldsymbol{b}_2 \;=\; \frac{1}{2}\begin{bmatrix} b_{n_s+1}+b_{n_s+2} \\ b_{n_s+2}-b_{n_s+1} \\ \\ \vdots \\ \\ b_{2n_s-1}+b_{2n_s} \\ b_{2n_s}-b_{2n_s-1} \end{bmatrix}\;, \tag{53}$$

$$F = T_2H_1 =$$

$$= -\frac{1}{2}\begin{bmatrix} A_4 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\ -A_4 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\ O & A_3 & A_4 & A_3 & -A_4 & \cdots & O & O & O & O & O \\ O & -A_3 & -A_4 & A_3 & -A_4 & \cdots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & A_3 & A_4 & A_3 & -A_4 & O \\ O & O & O & O & O & \cdots & -A_3 & -A_4 & A_3 & -A_4 & O \\ O & O & O & O & O & \cdots & O & O & A_3 & A_4 & -A_4 \\ O & O & O & O & O & \cdots & O & O & -A_3 & -A_4 & -A_4 \end{bmatrix}\;, \tag{54}$$

$$\bar{F} = T_1H_2 =$$

$$= -\frac{1}{2}\begin{bmatrix} 2A_3 & -2A_4 & O & O & \cdots & O & O & O & O \\ A_3 & A_4 & A_3 & -A_4 & \cdots & O & O & O & O \\ -A_3 & -A_4 & A_3 & -A_4 & \cdots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & A_3 & A_4 & A_3 & -A_4 \\ O & O & O & O & \cdots & -A_3 & -A_4 & A_3 & -A_4 \\ O & O & O & O & \cdots & O & O & 2A_3 & 2A_4 \end{bmatrix}\;. \tag{55}$$

Παρατηρούμε ότι ο πίνακας TA έχει περίπου διπλάσια απεξαρτημένα διαγώνια blocks από ότι ο πίνακας A. Επομένως το παραπάνω preconditionig του συστήματος **διπλασίασε** τον βαθμό παραλληλοποίησης της εφαρμογής μας.

Η επόμενη ενότητα παρουσιάζει την εφαρμογή της επαναληπτικής μεθόδου AOR στο σύστημα που έχει προκύψει καθώς επίσης και την κατασκευή του αντίστοιχου παράλληλου αλγορίθμου.

3.3 Παράλληλος AOR αλγόριθμος

Ανακαλώντας τη βασική σχέση (27) ορισμού της μεθόδου AOR, καθώς και τις σχέσεις (50,51,54,55) οι οποίες καθορίζουν την τελική μορφή $\mathrm{TA}\boldsymbol{x} = T\boldsymbol{b}$ του collocation συστήματος και θεωρώντας τη διάσπαση

$$TA = TD_A - TL_A - TU_A = \begin{bmatrix} D & O \\ O & \bar{D} \end{bmatrix} - \begin{bmatrix} O & O \\ F & O \end{bmatrix} - \begin{bmatrix} O & \bar{F} \\ O & O \end{bmatrix} \quad (56)$$

η μέθοδος AOR παίρνει την ακόλουθη μορφή

$$\begin{bmatrix} D & O \\ -rF & \bar{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R^{(m+1)} \\ \boldsymbol{x}_B^{(m+1)} \end{bmatrix} = \begin{bmatrix} (1-\omega)D & \omega\bar{F} \\ (\omega-r)F & (1-\omega)\bar{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R^{(m)} \\ \boldsymbol{x}_B^{(m)} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{b}}_R \\ \hat{\boldsymbol{b}}_B \end{bmatrix} \quad (57)$$

όπου $\hat{\boldsymbol{b}} = [\hat{\boldsymbol{b}}_R \ \hat{\boldsymbol{b}}_B]^T = \omega[\tilde{\boldsymbol{b}}_R \ \tilde{\boldsymbol{b}}_B]^T$.

Έτσι το παραπάνω σύστημα γράφεται σε μορφή εξισώσεων ως εξής

$$\begin{cases} D\boldsymbol{x}_R^{(m+1)} = (1-\omega)D\boldsymbol{x}_R^{(m)} + \omega\bar{F}\boldsymbol{x}_B^{(m)} + \hat{\boldsymbol{b}}_R \\ \bar{D}\boldsymbol{x}_B^{(m+1)} = (1-\omega)\bar{D}\boldsymbol{x}_B^{(m)} + F[(\omega-r)\boldsymbol{x}_R^{(m)} + r\boldsymbol{x}_R^{(m+1)}] + \hat{\boldsymbol{b}}_B \end{cases}, \quad (58)$$

ή, ισοδύναμα

$$\begin{cases} D\,\Delta\boldsymbol{x}_R = \omega\bar{F}\boldsymbol{x}_B^{(m)} + \hat{\boldsymbol{b}}_R \\ \bar{D}\,\Delta\boldsymbol{x}_B = F[(\omega-r)\boldsymbol{x}_R^{(m)} + r\boldsymbol{x}_R^{(m+1)}] + \hat{\boldsymbol{b}}_B \end{cases}, \quad (59)$$

όπου

$$\begin{cases} \Delta\boldsymbol{x}_R^{(m+1)} = \boldsymbol{x}_R^{(m+1)} - (1-\omega)\boldsymbol{x}_R^{(m)} \\ \Delta\boldsymbol{x}_B^{(m+1)} = \boldsymbol{x}_B^{(m+1)} - (1-\omega)\boldsymbol{x}_B^{(m)} \end{cases}, \quad (60)$$

Θεωρώντας την διάσπαση LU των πινάκων A_1 και A_2, μέσω της απαλοιφής Gauss με μερική οδήγηση, θα ισχύει

$$P_1A_1 = L_1U_1 \qquad P_2A_2 = L_2U_2 \quad (61)$$

όπου P_1 και P_2 είναι αντίστοιχα οι μεταθετικοί πίνακες οι οποίοι προκύπτουν από τη διαδικασία της οδήγησης, η σχέση (59) γράφεται στην παρακάτω μορφή παράλληλου αλγορίθμου

Παράλληλος Αλγόριθμος 1

for $j = 1$ *to* n_s *do in parallel*

S1a: $\boldsymbol{c}_j \leftarrow \hat{\boldsymbol{b}}_j + \frac{\omega}{2}[A_3 \boldsymbol{v}_j^{(m)} + A_4 \boldsymbol{w}_j^{(m)}]$

S2a: *Solve* $\mathcal{W} \boldsymbol{y}_j = \tilde{P} \boldsymbol{c}_j$

S3a: $\boldsymbol{x}_j^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_j^{(m)} + \boldsymbol{y}_j$

for $j = 1$ *to* n_s *do in parallel*

S1b: $\boldsymbol{c}_j \leftarrow \hat{\boldsymbol{b}}_j + A_4[\frac{\omega - r}{2} \boldsymbol{z}_j^{(m)} + \frac{r}{2} \boldsymbol{z}_j^{(m+1)}] - A_3[\frac{\omega - r}{2} \boldsymbol{d}_j^{(m)} + \frac{r}{2} \boldsymbol{d}_j^{(m+1)}]$

S2b: *Solve* $\hat{\mathcal{W}} \boldsymbol{y}_j = \hat{P} \boldsymbol{c}_j$

S3b: $\boldsymbol{x}_{n_s+j}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{n_s+j}^{(m)} + \boldsymbol{y}_j$

όπου

$$\boldsymbol{v}_j = \begin{cases} -2\,\boldsymbol{x}_{n_s+1}\,, & \text{αν} \quad j = 1 \\ -\boldsymbol{x}_{n_s+j-1} - \boldsymbol{x}_{n_s+j+1}\,, & \text{αν} \quad j = 2, 4, .., n_s - 2 \\ \boldsymbol{x}_{n_s+j-2} - \boldsymbol{x}_{n_s+j}\,, & \text{αν} \quad j = 3, 5, .., n_s - 1 \\ 2\,\boldsymbol{x}_{2n_s-1}\,, & \text{αν} \quad j = n_s \end{cases}, \tag{62}$$

$$\boldsymbol{w}_j = \begin{cases} 2\,\boldsymbol{x}_{n_s+2}\,, & \text{αν} \quad j = 1 \\ \boldsymbol{x}_{n_s+j+2} - \boldsymbol{x}_{n_s+j}, & \text{αν} \quad j = 2, 4, .., n_s - 2 \\ \boldsymbol{x}_{n_s+j+2} + \boldsymbol{x}_{n_s+j}, & \text{αν} \quad j = 3, 5, .., n_s - 1 \\ 2\,\boldsymbol{x}_{2n_s}\,, & \text{αν} \quad j = n_s \end{cases}, \tag{63}$$

$$\boldsymbol{d}_j = \begin{cases} \boldsymbol{x}_2\,, & \text{αν} \quad j = 1, 2 \\ \boldsymbol{x}_{j-1} + \boldsymbol{x}_{j+1}, & \text{αν} \quad j = 3, 5, .., n_s - 3 \\ \boldsymbol{x}_j - \boldsymbol{x}_{j-2}, & \text{αν} \quad j = 4, 6, .., n_s - 2 \\ \boldsymbol{x}_{n_s-2}, & \text{αν} \quad j = n_s - 1 \\ -\boldsymbol{x}_{n_s-2}\,, & \text{αν} \quad j = n_s \end{cases}, \tag{64}$$

$$\boldsymbol{z}_j = \begin{cases} \boldsymbol{x}_{j+2} - \boldsymbol{x}_j\,, & \text{αν} \quad j = 1,3,...,n_s-3 \\ \boldsymbol{x}_{j+1} + \boldsymbol{x}_{j-1}\,, & \text{αν} \quad j = 2,4,...,n_s-2 \\ \boldsymbol{x}_{n_s} - \boldsymbol{x}_{n_s-1}\,, & \text{αν} \quad j = n_s-1 \\ \boldsymbol{x}_{n_s} + \boldsymbol{x}_{n_s-1}\,, & \text{αν} \quad j = n_s \end{cases}\,, \tag{65}$$

$$\mathcal{W} = \begin{cases} L_2U_2, & \text{αν} \quad j = 1,3,...,n_s-1,n_s \\ L_1U_1, & \text{αν} \quad j = 2,4,...,n_s-2 \end{cases}\,, \tag{66}$$

$$\hat{\mathcal{W}} = \begin{cases} L_2U_2, & \text{αν} \quad j = 1,3,...,n_s-1 \\ L_1U_1, & \text{αν} \quad j = 2,4,...,n_s \end{cases}\,, \tag{67}$$

$$\tilde{P} = \begin{cases} P_2, & \text{αν} \quad j = 1,3,...,n_s-1,n_s \\ P_1, & \text{αν} \quad j = 2,4,...,n_s-2 \end{cases}\,, \tag{68}$$

$$\hat{P} = \begin{cases} P_1, & \text{αν} \quad j = 1,3,...,n_s-1,n_s \\ P_2, & \text{αν} \quad j = 2,4,...,n_s-2 \end{cases}\,. \tag{69}$$

Η αριθμητική μέθοδος, η οποία εφαρμόστηκε παραπάνω για την επίλυση του Dirichlet - Poisson προβλήματος, μπορεί να χρησιμοποιηθεί κατάλληλα και για την επίλυση άλλων ελλειπτικών προβλημάτων. Ειδικά για το Dirichlet - Poisson πρόβλημα υπάρχει επιπλέον μια σημαντική πληροφορία. Είναι διαθέσιμοι οι αντίστροφοι των υποπινάκων A_1 και A_2 μέσω κλειστών τύπων[20]. Αυτό έχει σαν αποτέλεσμα να είναι διαθέσιμοι, σύμφωνα με τις σχέσεις (50) και (51), οι αντίστροφοι των block διαγώνιων πινάκων D και $\bar{D}$. Έτσι η επαναληπτική μέθοδος της AOR στη σχέση (57) μπορεί να γραφεί στην ακόλουθη μορφή εξισώσεων

$$\left\{ \begin{array}{rcl} \boldsymbol{x}_R^{(m+1)} & = & (1-\omega)\boldsymbol{x}_R^{(m)} \; + \; \omega\mathcal{M}\boldsymbol{x}_B^{(m)} \; + \; D^{-1}\hat{b}_R \\ \boldsymbol{x}_B^{(m+1)} & = & (1-\omega)\boldsymbol{x}_B^{(m)} \; + \; \hat{\mathcal{M}}[(\omega-r)\boldsymbol{x}_B^{(m)} \; + \; r\boldsymbol{x}_R^{(m+1)}] \; + \; \bar{D}^{-1}\hat{b}_B \end{array} \right. , \qquad (70)$$

όπου

$$\mathcal{M} = D^{-1}\bar{F} =$$

$$= \; - \left[\begin{array}{ccccccccccc} 2P & -2Q & O & O & O & \cdots & O & O & O & O & O \\ S & R & S & -R & O & \cdots & O & O & O & O & O \\ -P & -Q & P & -Q & O & \cdots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & O & S & R & S & -R \\ O & O & O & O & O & \cdots & O & -P & -Q & P & -Q \\ O & O & O & O & O & \cdots & O & O & O & -2P & -2Q \end{array} \right] , \qquad (71)$$

$$\hat{\mathcal{M}} = \bar{D}^{-1}F =$$

$$= \; - \left[\begin{array}{ccccccccccc} R & S & -R & O & O & \cdots & O & O & O & O & O \\ -Q & P & -Q & O & O & \cdots & O & O & O & O & O \\ O & S & R & S & -R & \cdots & O & O & O & O & O \\ O & -P & -Q & P & -Q & \cdots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & S & R & S & -R & O \\ O & O & O & O & O & \cdots & -P & -Q & P & -Q & O \\ O & O & O & O & O & \cdots & O & O & S & R & -R \\ O & O & O & O & O & \cdots & O & O & -P & -Q & -Q \end{array} \right] \qquad (72)$$

με

$$P = {A_2}^{-1}A_3 \quad , \quad Q = {A_2}^{-1}A_4 \quad , \quad S = {A_1}^{-1}A_3 \quad , \quad R = {A_1}^{-1}A_4,$$

ή, ισοδύναμα

$$\left\{ \begin{array}{rcl} \Delta\boldsymbol{x}_R & = & \omega\mathcal{M}\boldsymbol{x}_B^{(m)} \; + \; \hat{\boldsymbol{b}}_R \\ \Delta\boldsymbol{x}_B & = & \hat{\mathcal{M}}[(\omega-r)\boldsymbol{x}_R^{(m)} \; + \; r\boldsymbol{x}_R^{(m+1)}] \; + \; \hat{\boldsymbol{b}}_B \end{array} \right. , \qquad (73)$$

όπου $\hat{\boldsymbol{b}}_R \leftarrow D^{-1}\hat{\boldsymbol{b}}_R$ και $\hat{\boldsymbol{b}}_B \leftarrow \bar{D}^{-1}\hat{\boldsymbol{b}}_R$.

Οι παραπάνω σχέσεις είναι δυνατόν να εκφραστούν με τον ακόλουθο παράλληλο αλγόριθμο

Παράλληλος Αλγόριθμος 2

$\underline{for} \quad j = 1 \quad \underline{to} \; ; \; n_s \quad \underline{do\ in\ parallel}$

$\underline{if} \quad j \quad \underline{odd}$

S1: $\underline{then} \quad \boldsymbol{x}_j^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_j^{(m)} - \frac{\omega}{2}[P\boldsymbol{v}_j^{(m)} - Q\boldsymbol{w}_j^{(m)}] + \hat{b}_j$

S2: $\underline{else} \quad \boldsymbol{x}_j^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_j^{(m)} - \frac{\omega}{2}[S\boldsymbol{w}_j^{(m)} + R\boldsymbol{v}_j^{(m)}] + \hat{b}_j$

$\underline{for} \quad j = 1 \quad \underline{to} \; ; \; n_s \quad \underline{do\ in\ parallel}$

$\underline{if} \quad j \quad \underline{odd}$

S3: $\underline{then} \quad \boldsymbol{x}_{j+n_s}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{j+n_s}^{(m)} - S\,[\frac{\omega-r}{2}\boldsymbol{y}_j^{(m)} + \frac{r}{2}\boldsymbol{y}_j^{(m+1)}] + + \hat{b}_{j+n_s} - R\,[\frac{\omega-r}{2}\boldsymbol{z}_j^{(m)} + \frac{r}{2}\boldsymbol{z}_j^{(m+1)}]$

S4: $\underline{else} \quad \boldsymbol{x}_{j+n_s}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{j+n_s}^{(m)} - P\,[\frac{\omega-r}{2}\boldsymbol{z}_j^{(m)} + \frac{r}{2}\boldsymbol{z}_j^{(m+1)}] + + \hat{b}_{j+n_s} + Q\,[\frac{\omega-r}{2}\boldsymbol{y}_j^{(m)} + \frac{r}{2}\boldsymbol{y}_j^{(m+1)}]$

όπου

$$\boldsymbol{v}_j = \begin{cases} 2\,\boldsymbol{x}_{n_s+1}\,, & \text{if } j = 1 \\ \boldsymbol{x}_{n_s+j} - \boldsymbol{x}_{n_s+j-2}\,, & \text{if } j = 2,4,..,n_s-2 \\ \boldsymbol{x}_{n_s+j} - \boldsymbol{x}_{n_s+j+2}\,, & \text{if } j = 3,5,..,n_s-1 \\ -2\,\boldsymbol{x}_{2n_s-1}\,, & \text{if } j = n_s \end{cases}, \tag{74}$$

$$\boldsymbol{w}_j \;=\; \begin{cases} 2\;\boldsymbol{x}_{n_s+2}\;, & \text{if}\;\; j \;=\; 1 \\ \boldsymbol{x}_{n_s+j-1} \;+\; \boldsymbol{x}_{n_s+j+1}, & \text{if}\;\; j \;=\; 2,3,..,n_s-1 \\ -2\;\boldsymbol{x}_{2n_s}\;, & \text{if}\;\; j \;=\; n_s \end{cases} \quad , \tag{75}$$

$$\boldsymbol{y}_j \;=\; \begin{cases} \boldsymbol{x}_2\;, & \text{if}\;\; j \;=\; 1 \\ \boldsymbol{x}_{j-1} \;+\; \boldsymbol{x}_{j+1}, & \text{if}\;\; j \;=\; 2,3,..,n_s-2 \\ \boldsymbol{x}_{n_s-2}\;, & \text{if}\;\; j \;=\; n_s-1 \\ \boldsymbol{x}_{n_s-1} \;+\; \boldsymbol{x}_{n_s}\;, & \text{if}\;\; j \;;=\; n_s \end{cases} \quad , \tag{76}$$

$$\boldsymbol{z}_j \;=\; \begin{cases} \boldsymbol{x}_1 \;-\; \boldsymbol{x}_3\;, & \text{if}\;\; j \;=\; 1 \\ \boldsymbol{x}_2\;, & \text{if}\;\; j \;=\; 2 \\ \boldsymbol{x}_j \;-\; \boldsymbol{x}_{j+2}, & \text{if}\;\; j \;=\; 3,5,..,n_s-3 \\ \boldsymbol{x}_j \;-\; \boldsymbol{x}_{j-2}, & \text{if}\;\; j \;=\; 4,6,..,n_s-2 \\ \boldsymbol{x}_{n_s-1} \;-\; \boldsymbol{x}_{n_s}\;, & \text{if}\;\; j \;=\; n_s-1 \\ -\;\boldsymbol{x}_{n_s-2}\;, & \text{if}\;\; j \;=\; n_s \end{cases} \quad . \tag{77}$$

Στις παρακάτω ενότητες παρουσιάζεται η υλοποίηση του πρώτου παράλληλου αλγορίθμου σε μια virtual παράλληλη μηχανή μέσω του PVM και του δεύτερου παράλληλου αλγορίθμου σε ένα Distributed Memory MIMD παράλληλο υπολογιστικό σύστημα.

3.4 Εφαρμογή σε Virtual Παράλληλη Μηχανή

Η ενότητα αυτή περιγράφει την υλοποίηση του πρώτου παράλληλου αλγορίθμου σε μια virtual παράλληλη μηχανή μέσω του PVM. Στην αρχή περιγράφονται τα τεχνικά χαρακτηριστικά της μηχανής αυτής και στη συνέχεια αναφέρεται το είδος της αρχιτεκτονικής. Ταυτόχρονα παρουσιάζεται η εφαρμογή του αλγορίθμου στην παράλληλη αυτή αρχιτεκτονική και παραθέτουμε τις μετρήσεις και τα συμπεράσματα από πειράματα που πραγματοποιήθηκαν.

Η παράλληλη μηχανή περιλαμβάνει τρία υπολογιστικά συστήματα διαφορετικών αρχιτεκτονικών. Τα τεχνικά χαρακτηριστικά των τριων hosts εμφανίζονται στον παρακάτω πίνακα

Hostname	PVM_ARCH	Type	Unix version	RAM Memory	Mflops[1]
Thalis	HPPA	HP827	HP-UX 9.00	32 MB	12
Euclid	HPPA	HP730	HP-UX 9.01	64 MB	24
Zenon	ALPHA	Alpha DEC	OSF 1.3	32 MB	56

Όπως είναι φανερό, τα τρία υπολογιστικά συστήματα έχουν διαφορετικές δυνατότητες, ενώ είναι συνδεμένα μεταξύ τους σε τοπικό δίκτυο τύπου thin EtherNet.

Η παράλληλη αρχιτεκτονική, η οποία χρησιμοποιήθηκε για την εφαρμογή του αλγορίθμου αναφέρεται σαν master - slave. Σύμφωνα με την αρχιτεκτονική αυτή ορισμένα tasks χρησιμοποιούνται για να ελέγχουν τα υπόλοιπα. Έτσι είναι απαραίτητη η κατασκευή δύο διαφορετικών προγραμμάτων, τα οποία θα υλοποιούν τις master και τις slave διαδικασίες.

Η master διδικασία είναι υπεύθυνη για την οργάνωση και επικοινωνία των δεδομένων (τμήματα $\boldsymbol{x}_i^{(m)}$ της λύσης $\boldsymbol{x}^{(m)}$) προς και από τις slave διαδικασίες, καθώς επίσης και τον υπολογισμό των βημάτων S3a και S3b. Αυτό συμβαίνει διότι το υπολογιστικό κόστος των βημάτων S3a και S3b είναι πολύ μικρότερο από το κόστος επικοινωνίας μέσω δικτύου. Οι slave διαδικασίες είναι αυτές, οι οποίες αναλαμβάνουν την διεκπεραίωση ολόκληρου του υπολογιστικού τμήματος του αλγορίθμου, δηλαδή των βημάτων S1a, S1b, S2a και S2b.

Στο σημείο αυτό θα πρέπει να αναφερθεί, ότι όλοι οι πίνακες και τα αρχικά διανύσματα $\boldsymbol{x}^{(0)}$ και $\hat{\boldsymbol{b}}$ είναι αποθηκευμένα στην κύρια μνήμη των hosts από την αρχή της διαδικασίας, ενώ οι παραγοντοποιήσεις των πινάκων A_1 και A_2 πραγματοποιούνται μια και δια παντός πριν την έναρξη της επαναληπτικής διαδικασίας. Έτσι κατά την διάρκεια των επαναληπτικών βημάτων εκτελούνται μόνο οι backward / forward αντικαταστάσεις.

1 Benchmark στο Εργαστήριο Εφαρμοσμένων Μαθηματικών & Η/Υ με Double precision Fortran σε BLAS Level 3

Στο παράρτημα 1 παρουσιάζονται οι κώδικες των δύο προγραμμάτων, τα οποία υλοποιούν τον παράλληλο αλγόριθμο. Κάθε host αναλαμβάνει την εκτέλεση ενός slave προγράμματος, ενώ ένας από αυτούς αναλαμβάνει επιπλέον και την εκτέλεση του master προγράμματος. Τα προγράμματα γράφτηκαν σε γλώσσα Fortran με απλή ακρίβεια και έγινε χρήση των βιβλιοθηκών Linpack και Blas level 3 μέσω της βιβλιοθήκης Lapack για την επίτευξη standard και αξιόπιστων μετρήσεων.

Οι τρεις παρακάτω πίνακες παρουσιάζουν τα αποτελέσματα για την επίλυση προβλήματος με διαμέριση $n_s = 120$.

Ποσοστό Υπολογισμών

	Αρχιτεκτονική I	Αρχιτεκτονική II	Αρχιτεκτονική III	Αρχιτεκτονική IV
Thalis	100%	37.5%	33%	15%
Euclid	-	62.5%	-	38%
Zenon	-	-	67%	47%
Χρόνος Βήματος σε secs	0.98	0.99	0.97	0.98

Ποσοστό Υπολογισμών

	Αρχιτεκτονική I	Αρχιτεκτονική II	Αρχιτεκτονική III	Αρχιτεκτονική IV
Thalis	-	37.5%	-	20%
Euclid	100%	62.5%	42%	33%
Zenon	-	-	58%	47%
Χρόνος Βήματος σε secs	0.65	0.62	0.62	0.63

Ποσοστό Υπολογισμών

	Αρχιτεκ-τονική I	Αρχιτεκ-τονική II	Αρχιτεκ-τονική III	Αρχιτεκ-τονική IV
Thalis	-	-	30%	25%
Euclid	-	43%	-	35%
Zenon	100%	57%	70%	40%
Χρόνος Βήματος σε secs	0.56	0.53	0.53	0.53

Σε κάθε πίνακα περιέχονται τα αποτελέσματα από τις τέσσερις δυνατές αρχιτεκτονικές, οι οποίες μπορούν να προκύψουν. Ο host του οποίου το όνομα εμφανίζεται με έντονα γράμματα εκτελεί το master πρόγραμμα. Οι χρόνοι αναφέρονται σε cpu-time του master για να υπολογίσει τη λύση σε ένα επαναληπτικό βήμα. Το ποσοστό συμμετοχής κάθε slave στον υπολογισμό της λύσης παρουσιάζεται για κάθε αρχιτεκτονική.

Όπως προκύπτει από τις μετρήσεις μας το δυνατότερο μηχάνημα αναλαμβάνει και την περισσότερη εργασία. Τα καλύτερα αποτελέσματα επιτυγχάνονται όταν επιλέξομε για master το γρηγορότερο μηχάνημα. Αυτό συμβαίνει γιατί τα δεδομένα του δικού του slave, δεν μεταφέρονται μέσω του δικτύου, αλλά εσωτερικά όπου η μεταφορά πραγματοποιείται ταχύτερα. Επίσης παρατηρούμε ότι τα καλύτερα αποτελέσματα σε μια αρχιτεκτονική δεν εμφανίζονται στην περίπτωση συμμετοχής όλων των υπολογιστικών συστημάτων. Αυτό συμβαίνει, γιατί τα αργά μηχανήματα προκαλούν καθυστέρηση στην εκτέλεση του προγράμματος.

Η κατάσταση του δικτύου επιρεάζει σημαντικά την απόδοση του αλγορίθμου και αυτό προκύπτει από την αύξηση του επικοινωνιακού κόστους. Για παράδειγμα, αν επιτρέψουμε τον υπολογισμό των βημάτων S3a και S3b από τους slaves, τότε ο παράλληλος χρόνος είναι μεγαλύτερος από το σειριακό στη συγκεκριμένη παράλληλη μηχανή.

Γίνεται λοιπόν σαφές ότι η επικοινωνία μέσω δικτύου παίζει σημαντικό ρόλο στην απόδοση της παράλληλης PVM μηχανής. Πιστεύουμε λοιπόν αν η συνδεσμολογία των μηχανών αυτών βασιστούν σε Thick ή Fiber Optics δίκτυο, τότε θα είχαμε καλλίτερα αποτελέσματα.

3.5 Εφαρμογή σε Transputer based Παράλληλη Μηχανή

Τα βασικά χαρακτηριστικά μιας Transputer based Παράλληλης Μηχανής στην οποία εφαρμόσαμε τον δεύτερο παράλληλο αλγόριθμο είναι τα παρακάτω :

- Η παράλληλη μηχανή αποτελείται από $\mathbf{P}$=16 επεξεργαστές του τύπου T800 και είναι συνδεμένοι μεταξύ τους σε δίκτυο.
- Το δίκτυο είναι σύγχρονο.
- Κάθε επεξεργαστής λαμβάνει δεδομένα χωρίς χρονική καθυστέριση.
- Κάθε επεξεργαστής διαθέτει δική του μνήμη των 4MB.

Για να εφαρμοστεί ο αλγόριθμος, όπως διατυπώθηκε παραπάνω, απαιτούνται n_s επεξεργαστές για την εκτέλεση όλων των παράλληλων διαδικασιών. Δηλαδή χρειάζεται απεριόριστος αριθμός επεξεργαστών και αυτό είναι εφικτό μόνο σε θεωρητικές μελέτες. Ετσι για την περίπτωση, κατά την οποία υπάρχει περιορισμένος αριθμός $\mathbf{P}$ επεξεργαστών με ($\mathbf{P} < n_s$), μπορούμε να θεωρήσουμε ότι $n_s = \mu\mathbf{P} + \upsilon \;\; (\mu, \upsilon \in \mathbb{N})$, οπότε οι παράλληλοι υπολογισμοί διαμερίζονται σε $\mu + 1$ ανεξάρτητα τμήματα. Καθένα από αυτά τα υπολογιστικά τμήματα απαρτίζεται από $\mathbf{P}$ παράλληλες διαδικασίες.

Ακολουθώντας την στρατηγική αυτή ο δεύτερος παράλληλος αλγόριθμος αναδιατυπώνεται ως εξής :

Παράλληλος Αλγόριθμος για fixed size Αρχιτεκτονικές

Part 1

$\mathbf{S} \leftarrow \mathbf{P}$

for $\kappa = 0$ *to* μ *do sequentially*

if $\kappa = \mu$ $\quad \mathbf{S} \leftarrow v$

for $j = 1$ *to* $\mathbf{S}$ *do in parallel*

if $j + \kappa\mathbf{P}$ *odd*

S1: *then* $\boldsymbol{x}_{j+\kappa\mathbf{P}}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{j+\kappa\mathbf{P}}^{(m)} - \frac{\omega}{2}[P\boldsymbol{v}_{j+\kappa\mathbf{P}}^{(m)} - Q\boldsymbol{w}_{j+\kappa\mathbf{P}}^{(m)}] + \hat{b}_{j+\kappa\mathbf{P}}$

S2: *else* $\boldsymbol{x}_{j+\kappa\mathbf{P}}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{j+\kappa\mathbf{P}}^{(m)} - \frac{\omega}{2}[S\boldsymbol{w}_{j+\kappa\mathbf{P}}^{(m)} + R\boldsymbol{v}_{j+\kappa\mathbf{P}}^{(m)}] + \hat{b}_{j+\kappa\mathbf{P}}$

Part 2

$\mathbf{S} \leftarrow \mathbf{P}$

for $\kappa = 0$ *to* μ *do sequentially*

if $\kappa = \mu$ $\quad \mathbf{S} \leftarrow v$

for $j = 1$ *to* $\mathbf{S}$ *do in parallel*

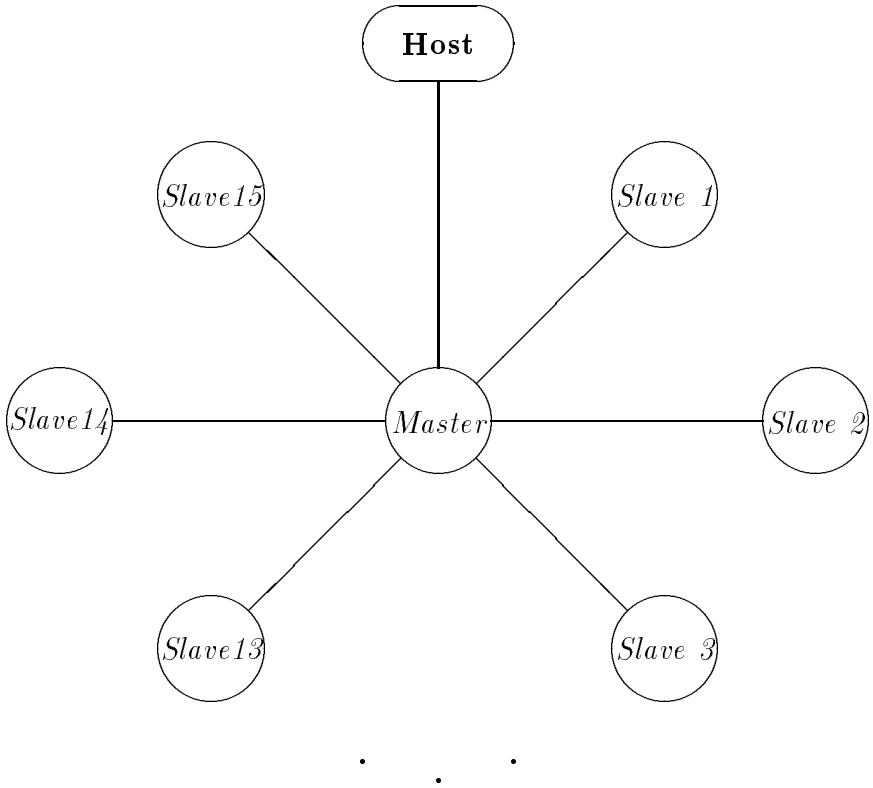
if $j + \kappa\mathbf{P}$ *odd*

S3: *then* $\boldsymbol{x}_{j+n_s+\kappa\mathbf{P}}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{j+n_s+\kappa\mathbf{P}}^{(m)} - S\,[\frac{\omega-r}{2}\boldsymbol{y}_{j+\kappa\mathbf{P}}^{(m)} + \frac{r}{2}\boldsymbol{y}_{j+\kappa\mathbf{P}}^{(m+1)}] +$

$+ \hat{b}_{j+n_s+\kappa\mathbf{P}} - R\,[\frac{\omega-r}{2}\boldsymbol{z}_{j+\kappa\mathbf{P}}^{(m)} + \frac{r}{2}\boldsymbol{z}_{j+\kappa\mathbf{P}}^{(m+1)}]$

S4: *else* $\boldsymbol{x}_{j+n_s+\kappa\mathbf{P}}^{(m+1)} \leftarrow (1-\omega)\boldsymbol{x}_{j+n_s+\kappa\mathbf{P}}^{(m)} - P\,[\frac{\omega-r}{2}\boldsymbol{z}_{j+\kappa\mathbf{P}}^{(m)} + \frac{r}{2}\boldsymbol{z}_{j+\kappa\mathbf{P}}^{(m+1)}]$

$+ \hat{b}_{j+n_s+\kappa\mathbf{P}} + Q\,[\frac{\omega-r}{2}\boldsymbol{y}_{j+\kappa\mathbf{P}}^{(m)} + \boldsymbol{y}_{j+\kappa\mathbf{P}}^{(m+1)}]$



Εικόνα 25 : Η Star Αρχιτεκτονική

Οι διάφορες μεταβλητές στον παραπάνω αλγόριθμο έχουν οριστεί στις σχέσεις (74-77). Στην υλοποίηση του αλγορίθμου έγινε χρήση της Star master - slave αρχιτεκτονικής, η οποία επιδεικνύεται σχηματικά στην Εικόνα 25.

Οι υπολογισμοί κάθε επαναληπτικού βήματος εμπλέκουν τους πίνακες P , Q, S και R μαζί με τα διανύσματα $\boldsymbol{x}$ και $\hat{\boldsymbol{b}}$. Έτσι αποθηκεύονται οι πίνακες στη μνήμη των slave επεξεργαστών, ενώ τα διανύσματα στη μνήμη του master.

Ειδικότερα στο m επαναληπτικό βήμα και για τη δημιουργία του $\boldsymbol{x}_i^{(m)}$ τμήματος της λύσης, ο master τροφοδοτεί έναν slave επεξεργαστή με τα κατάλληλα διανύσματα $\boldsymbol{x}_i^{(m)}$, $\boldsymbol{u}_i$, $\boldsymbol{w}_i$, $\boldsymbol{y}_i$, $\boldsymbol{z}_i$ και $\hat{\boldsymbol{b}}_i$. Ο slave επεξεργαστής εκτελεί τα βήματα S1 και S2 ή S3 και S4 υπολογίζοντας το $\boldsymbol{x}_i^{(m)}$ τμήμα της λύσης το οποίο στέλνει στο master. Δηλαδή ο master επεξεργαστής συντονίζει τη δημιουργία της λύσης, καθορίζοντας την διαμέρισή της τροφοδοτώντας και συλλέγοντας τα απαραίτητα δεδομένα σειριακά, ενώ οι slave επεξεργαστές έχουν αναλάβει το υπολογιστικό

μέρος του αλγορίθμου.

Η διαμέριση απαιτεί τα τμήματα της λύσης να έχουν τάξη $2n_s$, γι' αυτό τα διανύσματα $\boldsymbol{x}$ και $\hat{\boldsymbol{b}}$ αποθηκεύονται στη μνήμη του master επεξεργαστή σε μορφή διδιάστατων πινάκων μεγέθους $2n_s \times 2n_s$.
Παρατηρούμε ότι η τάξη των πινάκων είναι $O(n_s)$, ενώ η τάξη του συστήματος που επιλύεται είναι $O(n_s^2)$.

Για να μελετήσουμε την συμπεριφορά του αλγορίθμου, ακολουθήσαμε τις παρακάτω τεχνικές :

1. Για καθορισμένο μέγεθος του προβλήματος ($n_s = 60$) θεωρούμε το χρόνο σαν συνάρτηση του αριθμού των επεξεργαστών.

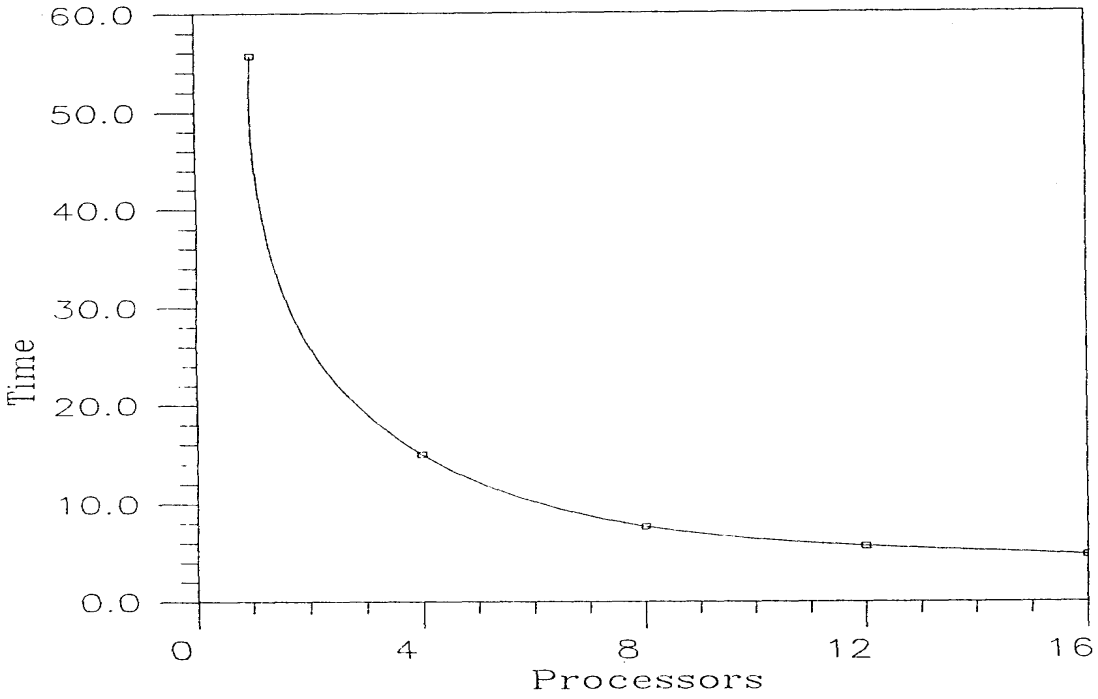
2. Για καθορισμένο αριθμό επεξεργαστών (**P**=15) θεωρούμε το χρόνο σαν συνάρτηση του μεγέθους του προβλήματος, για καθορισμό του speedup του παράλληλου αλγορίθμου.

Η Εικόνα 26 εμφανίζει τα αποτελέσματα από την πρώτη κατηγορία δοκιμών. Όπως προκύπτει από το γράφημα ο χρόνος σε κάθε επαναληπτικό βήμα μειώνεται σημαντικά καθώς αυξάνεται το πλήθος των επεξεργαστών. Τα αποτελέσματα από το δεύτερο είδος δοκιμών παρουσιάζονται στην Εικόνα 27.

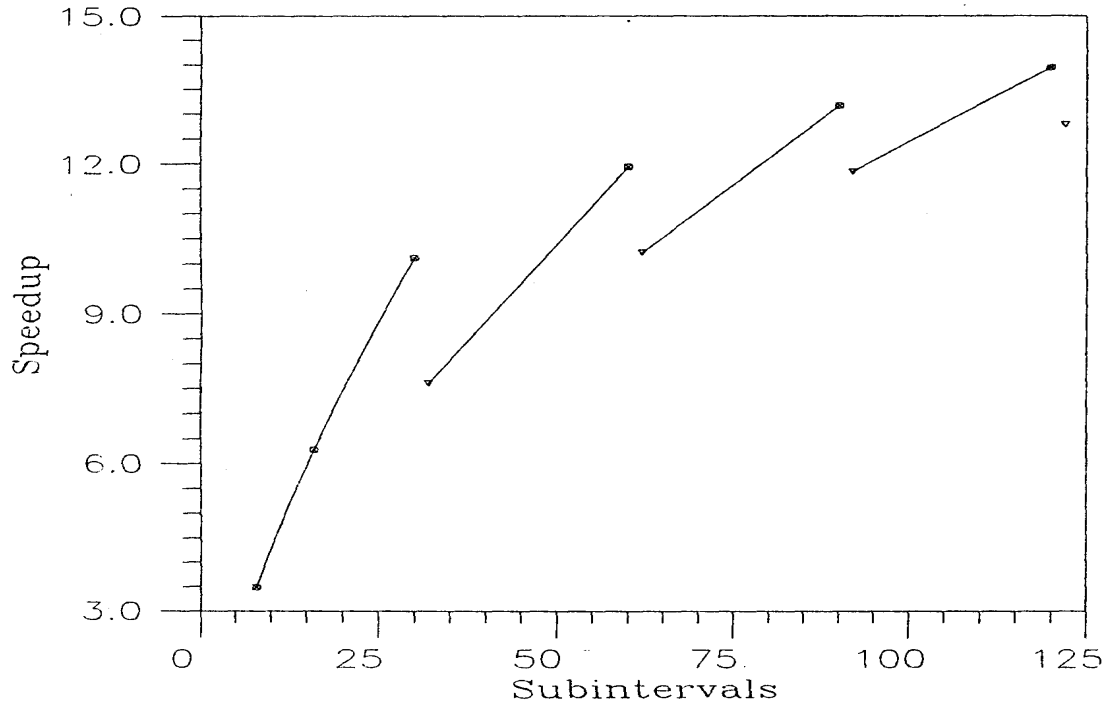
Το speedup, το οποίο ορίζεται να είναι

$$speedup = \frac{\text{Παράλληλος χρόνος}}{\text{Σειριακός χρόνος}} \quad ,$$

υπολογίστηκε για προβλήματα με αριθμό υποδιατημάτων από $n_s = 8$ μέχρι 122. Δηλαδή επιλύθηκαν γραμμικά συστήματα τάξης περίπου 60.000, ενώ ο πίνακας συντελεστών είχε περίπου 3.6×10^9 στοιχεία. Όπως είναι εμφανές η τιμή του speedup αυξάνει σύμφωνα με την αύξηση του μεγέθους του προβλήματος. Οι διακυμάνσεις, οι οποίες παρουσιάζονται στη μετάβαση από τις βέλτιστες περιπτώσεις με $n_s = k\mathbf{P}$ στις χειρότερες με $n_s = k\mathbf{P} + 2$ για $k = 1, 2, ...,$



Εικόνα 26 : Παράλληλος χρόνος για $n_s = 60$.



Εικόνα 27 : Μετρήσεις της τιμής του speedup.

οφείλονται στην διαμέριση των παράλληλων διαδικασιών. Για τις τιμές των βέλτιστων περιπτώσεων επιτυγχάνεται απόδοση μέχρι 90%.

Οι παραπάνω επιδόσεις του παράλληλου αλγορίθμου είναι συγκρίσιμες με τις αντίστοιχες των αλγορίθμων από τις εργασίες [16, 17] στην ίδια παράλληλη μηχανή. Έτσι τα σημαντικότερα πλεονεκτήματα της παρούσας υλοποίησης είναι τα εξής :

- Αύξηση του βαθμού παραλληλοποίησης
- Μεγαλύτερη ταχύτητα σύγκλισης

Πράγματι, με το precondition που εφαρμόστηκε στον collocation πίνακα, επιτεύχθηκε ο διπλασιασμός του πλήθους των διαγωνίων blocks και μαζί του βαθμού παραλληλοποίησης. Επίσης, διπλασιάστηκε το μέγεθος των προβλημάτων, τα οποία είναι δυνατό να επιλυθούν χωρίς επιπλέον υπολογιστικό κόστος.

Η ανάλυση της σύγκλισης, όπως παρουσιάζεται στην εργασία [17], αναφέρει δραματική μείωση του αριθμού των επαναλήψεων χρησιμοποιώντας την νέα αρίθμηση αγνώστων και εξισώσεων. Ενδεικτικά αναφέρουμε ότι στο πρόβλημα με $n_s = 16$, επιτεύχθηκε σύγκλιση σε 43 βήματα, ενώ με την block τριδιαγώνια αρίθμηση χρειάστηκαν 247 βήματα.

ΠΑΡΑΡΤΗΜΑ 1

Κώδικες Προγραμμάτων

Πρόγραμμα Master

Πρόγραμμα Slave

ΠΑΡΑΡΤΗΜΑ 2

Βιβλιογραφία

References

[1] A.Beguelin, J.J.Dongarra, G.A.Geist, R.Manchek and V.S.Sunderam. Graphical Tools for network-based concurrent supercomputing. In *Proceedings of Supercomputing 91*, pp 435-444, Albuquerque, 1991.

[2] A.Beguelin, J.Dongarra, A.Geist, R.Manchek, K.Moore, P.Newton and V.Sunderam. HeNCE: A Users' Guide.Oak Ridge National Laboratory - Univ. of Tennessee, 1994.

[3] B.P. Bertsekas and J.N. Tsitsiklis, "Parallel and Distributed Computation", *Prentice Hall*, New Jersey, 1989

[4] G. Birkiff, M. H. Schultz and R. S. Varga, "Piecewise Hermite in one and two Variables with Applications to Partial Differential Equations", *Numer. Math.*, **11**, 232-256 (1968).

[5] C. de Boor and Swartz, "Collocation at Gaussian Points", *Siam J Numer. Anal.* **10**, 582-606 (1973).

[6] J.Dongarra, G.A.Geist, R.Manchek, V.S.Sunderam. Integrated PVM Framework Supports Heterogeneous Network Computing. Oak Ridge National Laboratory - Univ. of Tennessee, 1993.

[7] J. Douglas and T. Dupont, *Collocation Methods for Parabolic Equations in a Single Space Variable*, Springer-Verlag Lecture Notes **385**, Berlin/New York,1974.

[8] A.Geist, A.Beguelin, J.Dongarra, W.Jiang, R.Manchek and V.Sunderam. PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Network Parallel Computing. MIT 1995.

[9] A.Geist, A.Beguelin, J.Dongarra, W.Jiang, R.Manchek and V.Sunderam. PVM 3 User's Guide and Reference Manual. Oak Ridge National Laboratory - Univ. of Tennessee, 1994.

[10] A. Hadjidimos, T.S. Papatheodorou and Y. G. Saridakis, "Optimal Block Iterative Schemes for Certain Large Sparse and Non-symmetric Linear Systems", *Linear Algebra Appl.*, **110**, 285-318 (1988).

[11] L. A. Hageman and D. M. Young, " Applied Iterative Methods", *Academic Press*,
New York, 1981

[12] E. N. Houstis, "Application of the method of Collocation on Lines for Solving Nonlinear Hyperbolic Problems",*Math. Comp.*, **31**, 443 - 456 (1977).

[13] E. N. Houstis, W. Mitchell, J. R. Rice, "Collocation Software for Second Order Elliptic Partial Differential Equations", *ACM Trans. Math. Software*, **11**, 379-412 (1985).

[14] E. N. Houstis, R. E. Lynch, T. S. Papatheodorou and J. R. Rice, "Evaluation of Numerical Methods for Elliptic Partial Differential Equations", *J Comp. Phys.*, **27**, 323 - 350 (1978).

[15] Yu-Lin Lai, A. Hadjidimos, E. N. Houstis and J. R. Rice, "On the iterative solution of Hermite Collocation Equations", CSD-TR-92-094, Purdue University, 1992.

[16] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Iterative PDE Computations on a Transputer based Parallel Computer", Procs *2nd Hellenic Conference on Mathematics and Informatics (HERMIS '94)*, 425-431, 1994.

[17] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Mapping Parallel Iterative Algorithms for PDE Computations on a Distributed Memory Computers ", *Parallel Algorithms and Applications* , **10**, (34).

[18] J.M. Ortega and R.G. Voigt, "Solution of Partial Differential Equations on Vector and Parallel Computers", *SIAM Review* **27***(2)*, 149-240, 1985

[19] E. Papadopoulou, Y.G. Saridakis and T.S. Papatheodorou, "Orderings and Partitions of PDE Computations for a Fixed Size VLSI Architecture", Procs *of IEEE-ACM Fall Joint Computer Science Conference*, 366-374, Dallas, 1987.

[20] T.S. Papatheodorou, "Inverses for a Class of Banded Matrices and Applications to Piecewise Cubic Approximation", *J. Comp. Appl. Math.* **8***(4)*, 285-288, 1982.

[21] T.S. Papatheodorou, "Block AOR Iteration for Nonsymmetric Matrices" , *Math. Comp.* **41***(164)*, 511-525, 1983

[22] Y.G. Saridakis, " Parallelism, Applicability and Optimality of Modern Iterative Methods", *Phd. Thesis*, Clarkson University, 1985.

[23] Y.G. Saridakis, "Optimally Repartitioned SOR Iterative Method for p-Cyclic Collocation Matrices",Procs *2nd Hellenic Conference on Mathematics and Info rmatics (HERMIS '94)*, 1994

[24] R.S. Varga, "Matrix Iterative Analysis",*Prentice Hall*, Englew ood Cliffs, NJ, 1962

[25] D.M. Young, "Iterative Solution of Large Linear Systems", *Academic Press*, New York, 1981