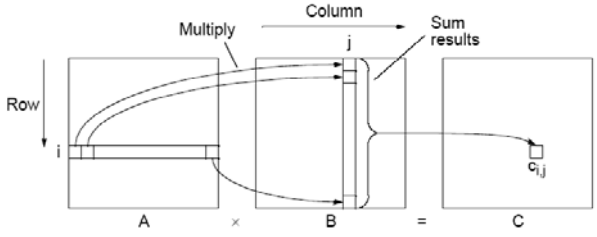
Αριθμητικοί Αλγόριθμοι

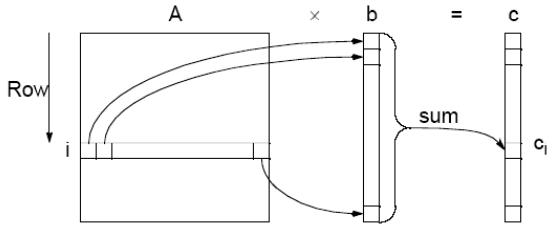
Άθροιση και Πολλαπλασιασμός Πινάκων

- Το άθροισμα C[nxn] δύο πινάκων A[nxn] και B[nxn] (C=A+B), προκύπτει από το άθροισμα των αντίστοιχων στοιχείων των A και B δηλ. $c_{i,j} = a_{i,j} + b_{i,j}$ ($0 \le i < n$, $0 \le j < n$)
- Το γινόμενο C[nxm] δύο πινάκων A[nxl] και B[lxm] ορίζεται ως εξής:

$$c_{i,j} = \sum_{k=0}^{l-1} a_{ik} b_{kj} \; i = 0,\ldots,n-1,\; j = 0,\ldots,m-1$$



- Ο πολλαπλασιασμός πίνακα με διάνυσμα προκύπτει από το προηγούμενο ορισμό του γινόμενου πινάκων αν θεωρήσουμε το B μονοδιάστατο πίνακα (B[lx1])



• Είναι επίσης γνωστό ότι ένα σύστημα γραμμικών εξισώσεων μπορεί να γραφεί υπό μορφή πινάκων ως Ax=b όπου x είναι το διάνυσμα των αγνώστων, A είναι ο πίνακας των συντελεστών των αγνώστων και b είναι το διάνυσμα των σταθερών των εξισώσεων.
• Ο ακολουθιακός κώδικας για τον πολλαπλασιασμό των πινάκων έχει ως εξής:

```
for (i = 0; i < n; i++)
                    for (j = 0; j < n; j++) {
                                c[i][j] = 0;
                                for (k = 0; k < n; k++)
                                            c[i][j] = c[i][j] + a[i][k] * b[k][j];
          }
```

• Ο αλγόριθμος αυτός απαιτεί n^3 πολλαπλασιασμούς και n^3 προσθέσεις και επομένως η πολυπλοκότητα του αλγορίθμου θα είναι $O(n^3)$.
• Ο παραπάνω υπολογισμός μπορεί εύκολα να παραλληλοποιηθεί αφού δεν υπάρχουν εξαρτήσεις μεταξύ των επαναλήψεων των δύο εξωτερικών βρόχων. Έτσι:
• Με n επεξεργαστές, κάθε επεξεργαστής μπορεί να αναλάβει μία επανάληψη του εξωτερικού βρόχου. Κάθε επεξεργαστής εκτελεί $O(n^2)$ πράξεις.
• Με n^2 επεξεργαστές, κάθε επεξεργαστής αναλαμβάνει τον υπολογισμό ενός στοιχείου του πίνακα C. Κάθε επεξεργαστής εκτελεί O(n) πράξεις.
• Με n^3 επεξεργαστές, μπορούμε να υλοποιήσουμε παράλληλα και τον υπολογισμό κάθε στοιχείου του πίνακα C. Σε αυτή την περίπτωση, η πολυπλοκότητα του παράλληλου αλγόριθμου είναι O(log n).
Μπορεί να αποδειχθεί ότι το κάτω όριο πολυπλοκότητας για τον παράλληλο πολλαπλασιασμό πινάκων είναι Ω(log n).
Πρέπει να σημειωθεί ότι οι παραπάνω πολυπλοκότητες δεν περιλαμβάνουν το χρόνο επικοινωνίας που απαιτεί αλλά αντιστοιχούν μόνο σε χρόνο επεξεργασίας.

Χωρισμός Πινάκων σε Υποπίνακες

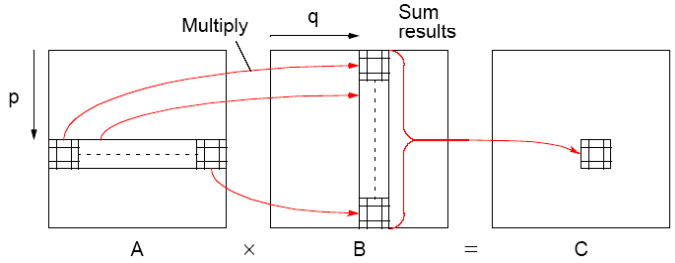
• Στη πράξη, έχουμε στη διάθεση μας πολύ λιγότερους από n επεξεργαστές. Σε αυτή τη περίπτωση κάθε επεξεργαστής αναλαμβάνει τον υπολογισμό περισσότερων στοιχείων του πίνακα C.
• Συγκεκριμένα, αν έχουμε s^2 επεξεργαστές, μπορούμε να χωρίζουμε κάθε πίνακα σε s^2 υποπίνακες διαστάσεων n/s x n/s elements. Αν $A_{p,q}$ είναι ο υποπίνακας στη γραμμή p και στη στήλη q του block πίνακα A, το γινόμενο C=AB μπορεί να υπολογισθεί ως εξής:

```
for (p = 0; p < s; p++)
  for (q = 0; q < s; q++) {
    Cp,q = 0;                          /* clear elements of submatrix */
    for (r = 0; r < m; r++)            /* submatrix multiplication &*/
      Cp,q = Cp,q + Ap,r * Br,q;       /*add to accum. submatrix*/
}
```

Αξίζει να σημειωθεί ότι στον υπολογισμό

```
Cp,q = Cp,q + Ap,r * Br,q;
```

όλες οι πράξεις είναι μεταξύ πινάκων διαστάσεων n/s x n/s. Αυτός ο τρόπος πολλαπλασιασμού πινάκων είναι γνωστός ως πολλαπλασιασμός block πινάκων



Για παράδειγμα για τον υπολογισμό του γινομένου δύο πινάκων Α και Β διαστασεων 4x4, μπορούμε να χωρίσουμε τους πίνακες σε υποπίνακες 2x2 και στη συνέχεια να εκτελέσουμε τον υπολογισμό σε block πίνακες:

$$\left[\begin{array}{cc|cc} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{array}\right] \times \left[\begin{array}{cc|cc} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{array}\right]$$

Έτσι ο αρχικός υπολογισμός ανάγεται ουσιαστικά στη πρόσθεση και πολλαπλασιασμό πινάκων 2x2. Π.χ. ο υπολογισμός του στοιχείου C_{00} σύμφωνα με τον τύπο $C_{00}=A_{00}B_{00}+A_{01}B_{10}$ συνεπάγεται στους ακόλουθους υπολογισμούς:

$$\overset{A_{0,0}}{\begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{bmatrix}} \times \overset{B_{0,0}}{\begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix}} + \overset{A_{0,1}}{\begin{bmatrix} a_{0,2} & a_{0,3} \\ a_{1,2} & a_{1,3} \end{bmatrix}} \times \overset{B_{1,0}}{\begin{bmatrix} b_{2,0} & b_{2,1} \\ b_{3,0} & b_{3,1} \end{bmatrix}}$$

$$= \begin{bmatrix} a_{0,0}b_{0,0}+a_{0,1}b_{1,0} & a_{0,0}b_{0,1}+a_{0,1}b_{1,1} \\ a_{1,0}b_{0,0}+a_{1,1}b_{1,0} & a_{1,0}b_{0,1}+a_{1,1}b_{1,1} \end{bmatrix} + \begin{bmatrix} a_{0,2}b_{2,0}+a_{0,3}b_{3,0} & a_{0,2}b_{2,1}+a_{0,3}b_{3,1} \\ a_{1,2}b_{2,0}+a_{1,3}b_{3,0} & a_{1,2}b_{2,1}+a_{1,3}b_{3,1} \end{bmatrix}$$

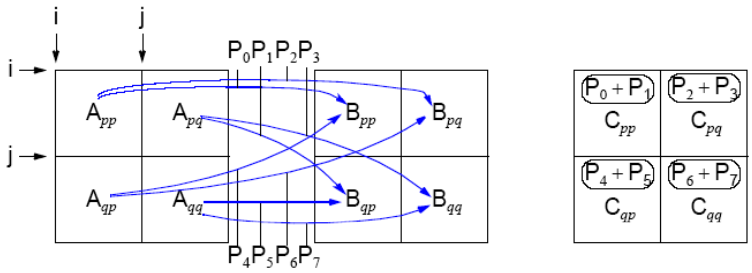
$$= \begin{bmatrix} a_{0,0}b_{0,0}+a_{0,1}b_{1,0}+a_{0,2}b_{2,0}+a_{0,3}b_{3,0} & a_{0,0}b_{0,1}+a_{0,1}b_{1,1}+a_{0,2}b_{2,1}+a_{0,3}b_{3,1} \\ a_{1,0}b_{0,0}+a_{1,1}b_{1,0}+a_{1,2}b_{2,0}+a_{1,3}b_{3,0} & a_{1,0}b_{0,1}+a_{1,1}b_{1,1}+a_{1,2}b_{2,1}+a_{1,3}b_{3,1} \end{bmatrix}$$

$$= C_{0,0}$$

• Στη παράλληλη υλοποίηση του πολλαπλασιασμού πινάκων, θεωρούμε ότι υπάρχουν s^2 διεργασίες/επεξεργαστές $P_{i,j}$ (i,j=0,...,s-1) και κάθε διεργασία αναλαμβάνει τον υπολογισμό του υποπίνακα C_{ij} του πίνακα C.
• Για τον υπολογισμό αυτό, χρειάζεται τους πίνακες A_{ik} και B_{kj} (k=0,...,s-1) αφού ο πίνακας C_{ij} δίνεται από τη σχέση $C_{ij} = A_{i0}B_{0j} + A_{i1}B_{1j} +...+A_{i,s-1}B_{s-1,j}$.
• Ο υπολογισμός αυτός απαιτεί s πολλαπλασιασμούς πινάκων διαστάσεων n/s x n/s. Κάθε τέτοιος πολλαπλασιασμός πινάκων εκτελεί $(n/s)^3$ απλούς πολλαπλασιασμούς και επομένως ο υπολογισμός του υποπίνακα C_{ij} απαιτεί $s^* (n/s)^3 =n^3/s^2$ απλούς πολλαπλασιασμούς.
• Όσον αφορά το χρόνο επικοινωνίας, θεωρούμε ότι και οι δύο πίνακες Α και Β είναι αρχικά αποθηκευμένοι στη master διεργασία. Στη συνέχεια τα στοιχεία αυτά θα πρέπει να μοιραστούν στις s^2 slave διεργασίες. Με δύο διαφορετικά μηνύματα από τη master διεργασία, κάθε slave διεργασία θα λάβει $s^*(n/s)^2$ στοιχεία από τον πίνακα Α και $s^*(n/s)^2$ στοιχεία από το πίνακα Β, συνολικά $2n^2/s$ στοιχεία.
• Μετά τον υπολογισμό των υποπινάκων C_{ij} , τα $(n/s)^2$ στοιχεία κάθε υποπίνακα θα πρέπει να σταλούν πίσω στη master διεργασία. Έτσι συνολικά ο χρόνος επικοινωνίας θα είναι $T_{comm} = 2s^2 t_{startup} + 2n^2s t_{data} + s^2t_{startup} + n^2 t_{data} = 3s^2 t_{startup} + (2s+1) n^2t_{data}$

Αναδρομική υλοποίηση του πολλαπλασιασμού πινάκων

Όπως αναφέρθηκε προηγουμένως, ο πολλαπλασιασμός δύο πινάκων nxn μπορεί να αναχθεί σε πολλάπλασιασμό και πρόσθεση πινάκων n/s x n/s αν διαιρέσουμε κάθε πίνακα σε sxs υποπίνακες.
Αυτή ιδέα μπορεί να εφαρμοσθεί αναδρομικά με μία τεχνική διαίρει και βασίλευε.
Αρχικά, οι δύο nxn πίνακες A και B χωρίζονται σε 4 υποπίνακες. θεωρούμε ότι η διάσταση n είναι δύναμη του δύο.



Ο υπολογισμός ανάγεται στον υπολογισμό των 8 γινομένων $A_{ik}B_{kj}$ όπου i,j,k=p,q. Οι πίνακες σε αυτά τα γινόμενα είναι διαστάσεων n/4xn/4 και μπορούν να υπολογιστούν με την ίδια λογική, χωρίζοντας τους πίνακες αυτούς σε υποπίνακες n/8xn/8. Ακολουθώντας αυτή την αναδρομική λογική, θα φτάσουμε σε πίνακες 1x1 που ο πολλαπλασιασμός μπορεί να υλοποιηθεί με προφανή τρόπο.

Αναδρομικός αλγόριθμος

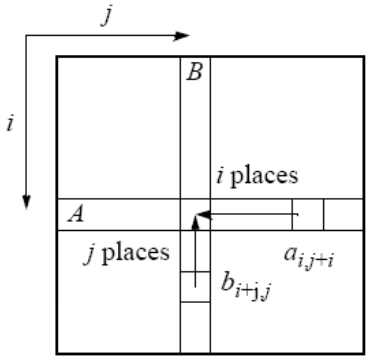
• Ο αναδρομικός αλγόριθμος για τον πολλαπλασιασμό πινάκων μπορεί να διατυπωθεί ως εξής:

```
mat_mult(App, Bpp, s)
{
if (s == 1)          /* if submatrix has one element */
   C = A * B;        /* multiply elements */
else {               /* continue to make recursive calls */
   s = s/2;          /* no of elements in each row/column */
   P0 = mat_mult(App, Bpp, s);
   P1 = mat_mult(Apq, Bqp, s);
   P2 = mat_mult(App, Bpq, s);
   P3 = mat_mult(Apq, Bqq, s);
   P4 = mat_mult(Aqp, Bpp, s);
   P5 = mat_mult(Aqq, Bqp, s);
   P6 = mat_mult(Aqp, Bpq, s);
   P7 = mat_mult(Aqq, Bqq, s);
   Cpp = P0 + P1;      /* add submatrix products to */
   Cpq = P2 + P3;      /* form submatrices of final matrix */
   Cqp = P4 + P5;
   Cqq = P6 + P7;
}
return (C);          /* return final matrix */
}
```

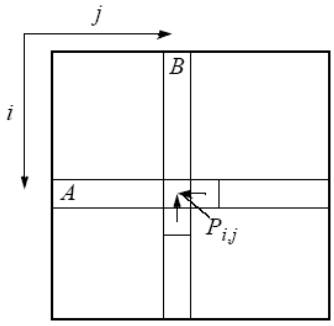
• Αυτός ο αναδρομικός υπολογισμός του γινομένου δύο πινάκων μπορεί εύκολα να υλοποιηθεί παράλληλα.
• Συγκεκριμένα, κάθε διεργασία/επεξεργαστής μπορεί να αναλάβει τον υπολογισμό μίας από τις οκτώ αναδρομικές κλήσεις της συνάρτησης.
• Είναι επίσης δυνατόν κάθε αναδρομική κλήση να εκτελεσθεί από περισσότερες από μία διεργασίες.
• Γενικά, μπορούμε να συνεχίσουμε την εκτέλεση της αναδρομής μέχρι το πλήθος των αναδρομικών κλήσεων να φθάσει το πλήθος των διαθέσιμων επεξεργαστών/διεργασιών.
• Ιδανικά, το πλήθος των διαθέσιμων διεργασιών θα πρέπει να είναι δύναμη του 8.
• Ο αλγόριθμος αυτός είναι κατάλληλος για υλοποίηση σε συστήματα κατανεμημένης μνήμης αφού καθώς προχωράει η αναδρομή οι προσβάσεις στα δεδομένα του πίνακα A και B έχουν τοπικό χαρακτήρα και επομένως οι περισσότερες από αυτές τις προσβάσεις ικανοποιούνται από τις τοπικές μνήμες cache των επεξεργαστών.

Ο αλγόριθμος του Cannon για πολλαπλασιασμό πινάκων

• Ο αλγόριθμος υποθέτει ότι το διασυνδετικό δίκτυο μεταξύ των επεξεργαστών είναι κυκλικό πλέγμα (torus). Η διάταξη του torus μπορεί εύκολα να προσομοιωθεί και σε άλλα διασυνδετικά δίκτυα. Π.χ. στο MPI μπορούμε εύκολα να προγραμματίσουμε την τοπολογία mesh.
• Στη πράξη όμως η υλοποίηση του πλέγματος σε μία αρχιτεκτονική cluster όπου όλοι οι υπολογιστές επικοινωνούν μέσω ενός κοινού διαύλου (π.χ. Ethernet) οδηγεί σε συμφόρηση στο κοινό μέσο λόγω των ταυτόχρονων μεταδόσεων που υπαγορεύει η αρχιτεκτονική του δικτύου mesh.
• Σε αυτή την περίπτωση το μέγεθος των υποπινάκων που χρησιμοποιεί ο αλγόριθμος Cannon πρέπει να είναι σχετικά μεγάλο, έτσι ώστε ο όγκος επεξεργασίας που εκτελεί κάθε επεξεργαστής μεταξύ διαδοχικών επικοινωνιών να αντισταθμίζει την επιβάρυνση λόγω επικοινωνιών.
Θα περιγράψουμε τον αλγόριθμο θεωρώντας ότι έχουμε στη διάθεση μας n^2 επεξεργαστές όσα και τα στοιχεία των πινάκων A, B και C. Όταν έχουμε λιγότερους επεξεργαστές , η διαδικασία που ακολουθεί μπορεί εύκολα να προσαρμοσθεί μόνο που αντί για απλά στοιχεία, η επεξεργασία γίνεται σε υποπίνακες των πινάκων A,B και C.
1. Αρχικά ο επεξεργαστής $P_{i,j}$ έχει τα στοιχεία $a_{i,j}$ και $b_{i,j}$ ($0 \le i < n$, $0 \le k < n$).
2. Τα στοιχεία των πινάκων A και B μετακινούνται κατά τέτοιο τρόπο έτσι ώστε στοιχεία των A και B που μπορούν να πολλαπλασιαστούν μεταξύ τους να βρεθούν στον ίδιο επεξεργαστή. Συγεκριμένα, η i-οστη γραμμή του A ολισθαίνει κατά i θέσεις αριστερά και η j-οστή στήλη του B ολισθαίνει j θέσεις προς τα πάνω. Αυτό έχει ως αποτέλεσμα τα στοιχεία $a_{i,j+i}$, και $b_{i+j,j}$ να βρεθούν στον επεξεργαστή $P_{i,j}$. Αυτά τα στοιχεία είναι χρήσιμα για τον υπολογισμό του στοιχείου $c_{i,j}$ του πίνακα C.
3. Κάθε επεξεργαστής $P_{i,j}$ πολλαπλασιάζει τα στοιχεία που έχει τοπικά.
4. Η i-οστή γραμμή του A ολισθαίνει μία θέση προς τα αριστερά και η j-οστη στήλη του B ολισθαίνει μία θέση προς τα πάνω. Αυτό έχει ως αποτέλεσμα κάθε επεξεργαστής να έχει δυο νέα στοιχεία των A και B που μπορούν να πολλαπλασιασθούν μεταξύ τους.
5. Κάθε επεξεργαστής $P_{i,j}$ πολλαπλασιάζει αυτά τα στοιχεία και το αποτέλεσμα προστίθεται στη τρέχουσα τιμή του στοιχείου $c_{i,j}$
6. Τα βήματα 4 και 5 επαναλαμβάνονται μέχρι το τελικό αποτέλεσμα να προκύψει (n - 1 οριζόντιες και κατακόρυφες ολισθήσεις απαιτούνται όταν οι πίνακες είναι nxn).



Βήμα 2 – Ευθυγράμμιση των στοιχείων A και B



Βήμα 4 – Ολίσθηση των στοιχείων των A και B κατά μία θέση

Θα αναλύσουμε την πολυπλοκότητα του αλγορίθμου όταν έχουμε sxs επεξεργαστές :

Επικοινωνία: Αρχική ευθυγράμμιση απαιτεί s-1 ολισθήσεις των στοιχείων των A και B. Κατά την εκτέλεση του αλγορίθμου θα έχουμε άλλες s-1 ολισθήσεις των στοιχείων των A και B. Κάθε ολίσθηση μεταφέρει $(n/s)^2$ στοιχεία. Άρα συνολικά θα έχουμε:

$$t_{comm} = 4(s-1)(t_{startup} + (n/s)^2 t_{data})$$

Υπολογισμός: Κάθε πολλαπλασιασμός και άθροιση υποπινάκων απαιτεί $(n/s)^3$ πολλαπλασιασμούς και αθροίσεις. Συνολικά θα έχουμε

$T_{comp} = 2s\,(n/s)^3 = 2n^3/\,s^2$

Επίλυση συστήματος γραμμικών εξισώσεων

• Έστω ένα σύστημα γραμμικών εξισώσεων:

$$
\begin{aligned}
a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 \quad &\cdots \quad + a_{n-1,n-1}x_{n-1} = b_{n-1} \\
&\vdots \\
a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 \quad &\cdots \quad + a_{2,n-1}x_{n-1} = b_2 \\
a_{1,0}x_0 + a_{1,1}x_1 + a_{1,2}x_2 \quad &\cdots \quad + a_{1,n-1}x_{n-1} = b_1 \\
a_{0,0}x_0 + a_{0,1}x_1 + a_{0,2}x_2 \quad &\cdots \quad + a_{0,n-1}x_{n-1} = b_0
\end{aligned}
$$

το οποίο σε μορφή πινάκων γράφεται ως Ax=b

• Ο στόχος είναι η εύρεση τιμών για τους αγνώστους x_0, x_1, …, x_{n-1}, με δεδομένες τιμές των $a_{0,0}$, $a_{0,1}$, …, $a_{n-1,n-1}$ και b_0, …, b_n.

• Πρώτα θα δούμε τρόπους επίλυσης του γραμμικού συστήματος όταν ο πίνακας Α είναι πυκνός δηλ. όταν τα περισσότερα στοιχεία του είναι μη μηδενικά.

• Στη συνέχεια, θα δούμε μεθόδους επίλυσης του γραμμικού συστήματος όταν ο πίνακας Α είναι αραιός.

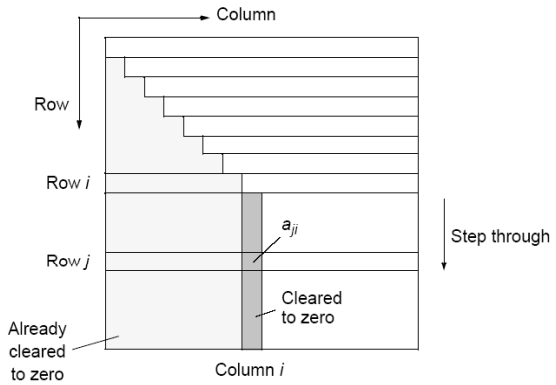
Επίλυση γραμμικού συστήματος με τη μέθοδο Gaussian Elimination

• Ο στόχος αυτής της μεθόδου είναι η μετατροπή ενός γενικού συστήματος γραμμικών εξισώσεων σε ένα τριγωνικό σύστημα εξισώσεων. Στη συνέχεια, το σύστημα αυτό επιλύεται με τη προς τα πίσω αντικατάσταση των αγνώστων (Back Substitution).

• Βασίζεται στο γεγονός ότι η τελική λύση ενός γραμμικού συστήματος δεν επηρεάζεται όταν σε οποιαδήποτε γραμμή προσθέσουμε κάποια άλλη γραμμή πολλαπλασιασμένη με μία σταθερά.

• Η διαδικασία αρχίζει από τη πρώτη γραμμή και επισκέπτεται κάθε γραμμή μέχρι και την τελευταία. Κατά την επίσκεψη της i-οστής γραμμής, αντικαθιστούμε κάθε γραμμή j κάτω από την i-οστή γραμμή με τη (γραμμή j) + (γραμμή i) $a_{j,i}/a_{i,i}$. Επομένως, μετά τις παραπάνω αντικαταστάσεις, όλα τα στοιχεία του πίνακα Α που είναι επί της j-οστής στήλης και κάτω από την i-οστή γραμμή μηδενίζονται αφού

$$a_{j,i} = a_{j,i} + a_{i,i}\left(\frac{-a_{j,i}}{a_{i,i}}\right) = 0$$



• Στο σχήμα φαίνεται ο μηδενισμός αυτών των στοιχείων κατά την επίσκεψη στην i-οστή γραμμή του πίνακα Α.

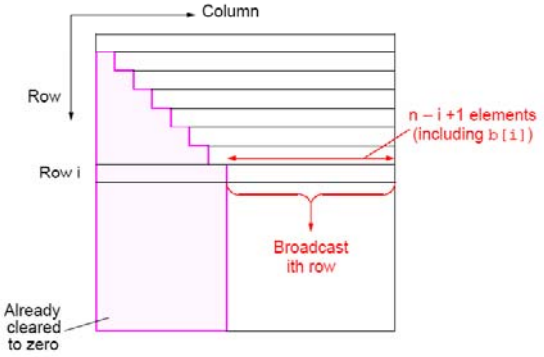
• Επίσης όλα τα στοιχεία που είναι σε στήλες πιο αριστερά της στήλης j και κάτω από τη διαγώνιο, έχουν ήδη μηδενιστεί από επισκέψεις σε προηγούμενες γραμμές του πίνακα.

• Αν κατά την επίσκεψη στην i-οστή γραμμή, το στοιχείο $a_{i,i}$ είναι μηδέν ή κοντά στο μηδέν, δεν θα μπορέσουμε να υπολογίσουμε την ποσότητα $-a_{j,i}/a_{i,i}$.
• Η διαδικασία θα πρέπει να τροποποιηθεί εφαρμόζοντας την τεχνική partial pivoting. Συγκεκριμένα το πρόβλημα επιλύεται με την αντιμετάθεση της i-οστής γραμμής με εκείνη τη γραμμή που είναι κάτω από την i και η οποία έχει το μεγαλύτερο σε απόλυτη τιμή στοιχείο στη στήλη j. Αξίζει να σημειωθεί ότι η επαναδιάταξη εξισώσεων δεν επηρεάζει τη λύση του γραμμικού συστήματος.
• Ο ακολουθιακός κώδικας για τη τεχνική Gaussian elimination χωρίς partial pivoting είναι ο εξής:

```
for (i = 0; i < n-1; i++)              /* for each row, except last */
        for (j = i+1; j < n; j++) {    /*step thro subsequent rows */
            m = a[j][i]/a[i][i];         /* Compute multiplier */
            for (k = i; k < n; k++)     /*last n-i-1 elements of row j*/
              a[j][k] = a[j][k] - a[i][k] * m;
            b[j] = b[j] - b[i] * m;         /* modify right side */
}
```

• Η συνολική πολυπλοκότητα του παραπάνω αλγορίθμου είναι $O(n^3)$.

Παράλληλη υλοποίηση



• Ο εξωτερικός βρόχος στο προηγούμενο πρόγραμμα είναι δύσκολο να παραλληλοποιηθεί γιατί η επεξεργασία σε κάθε επανάληψη εξαρτάται από τα αποτελέσματα όλων των προηγούμενων.
• Αντίθετα, οι επαναλήψεις του εσωτερικού βρόχου, μπορούν να παραλληλοποιηθούν αφού κάθε επανάληψη περιλαμβάνει υπολογισμούς σε διαφορετικές γραμμές του πίνακα.
• Έτσι αν έχουμε n επεξεργαστές, κάθε επεξεργαστής μπορεί να αναλάβει από μία γραμμή του πίνακα. Συγκεκριμένα ο επεξεργαστής P_i αναλαμβάνει την i-οστή εξίσωση
• Κατά την i-οστή επανάληψη του εξωτερικού βρόχου, θα πρέπει η διεργασία P_i (που έχει αναλάβει την i-οστη γραμμή) να στείλει τα περιεχόμενα της i-οστής εξίσωσης σε όλους τους επεξεργαστές P_j όπου j>i.
• Στη συνέχεια κάθε επεξεργαστής j>i, υπολογίζει τοπικά τις νέες τιμές των στοιχείων της i-οστής εξίσωσης.
• Είναι επίσης φανερό ότι όλοι οι επεξεργαστές P_j με j<i είναι αδρανείς κατά την i-οστή επανάληψη

• Στη συνέχεια θα προσδιορίσουμε την πολυπλοκότητα της παράλληλης υλοποίησης

Επικοινωνία : Συνολικά εκτελούνται n-1 λειτουργίες εκπομπής. Στην i-οστή λειτουργία, ο επεξεργαστής P_i στέλνει n-i+1 στοιχεία της i-οστής γραμμής στις υπόλοιπες διεργασίες (i=1…n). Συνολικά θα έχουμε:

$$t_{comm} = \sum_{i=1}^{n-1}\left(t_{startup} + (n-i+1)t_{data}\right) = (n-1)t_{startup} + \left(\frac{(n+2)(n+1)}{2} - 3\right)t_{data} = \mathrm{O}(n^2)$$

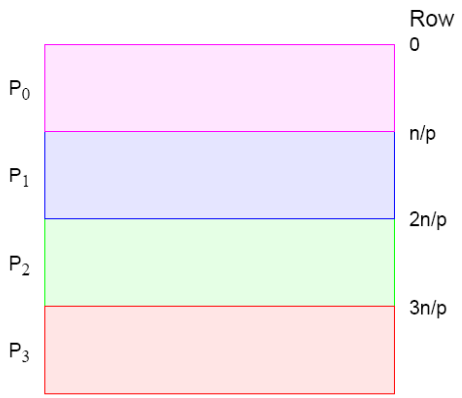
Επικοινωνία: Μετά από την λήψη των στοιχείων της γραμμής i που έστειλε ο επεξεργαστής P_i , κάθε επεξεργαστής P_j (j>i) εκτελεί *n - j* + 2 πολλαπλασιασμούς και *n - j* + 2 αφαιρέσεις για τον υπολογισμό των νέων τιμών της j-οστής γραμμής. Συνολικά θα έχουμε

$$t_{comp} = 2\sum_{i=1}^{n-1}(n-j+2) = \frac{(n+2)(n+1)}{2} - 3 = \mathrm{O}(n^2)$$

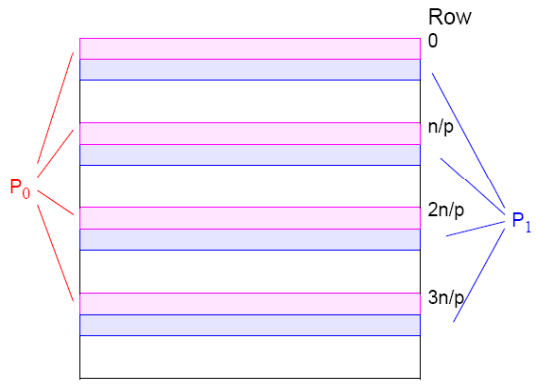
• Το βασικό πρόβλημα με αυτή την υλοποίηση είναι ότι προοδευτικά όλο και περισσότεροι επεξεργαστές είναι αδρανείς. Συγκεκριμένα, αν η τρέχουσα επανάληψη είναι στη i-οστή γραμμή, όλοι οι επεξεργαστές P_j (j<i) είναι αδρανείς.

• Στη πράξη, το πλήθος των διαθέσιμων επεξεργαστών είναι πολύ λιγότερο από πλήθος n των εξισώσεων. Σε αυτή την περίπτωση κάθε επεξεργαστής αναλαμβάνει περισσότερες από μία εξισώσεις (γραμμές) του γραμμικού συστήματος. Υπάρχουν δύο εναλλακτικές:

Κατανομή κατά λωρίδες (Strip allocation): Ο επεξεργαστής P_i αναλαμβάνει τις γραμμές i*(n/p)...(i+1)*(n/p)-1. Η τεχνική αυτή έχει πάλι το μειονέκτημα του προοδευτικά αυξανόμενου αριθμού επεξεργαστών που είναι αδρανείς.



Strip allocation



Cyclic-Striped Partitioning

Κατανομή κατά λωρίδες (Strip allocation): Ο επεξεργαστής P_i αναλαμβάνει τις γραμμές i+(n/p), i+2*(n/p), i+3*(n/p) .... Η κατανομή αυτή επιλύει το μειονέκτημα της προηγούμενης κατανομής και επιτυγχάνει καλύτερη κατανομή φόρτου στους επεξεργαστές.

Επαναληπτικές μέθοδοι επίλυσης γραμμικών συστημάτων

• Με χρήση *N* επεξεργαστών, η χρονική πολυπλοκότητα των απευθείας μεθόδων είναι $O(N^2)$, η οποία είναι σχετικά υψηλή.
• Η χρονική πολυπλοκότητα μίας επαναληπτικής μεθόδου εξαρτάται από διάφορους παράγοντες όπως:
  • τύπος επαναληπτικής μεθόδου,
  • πλήθος επαναλήψεων
  • πλήθος αγνώστων και
  • απαιτούμενη ακρίβεια
• Οι επαναληπτικές μέθοδοι είναι προτιμότερες όταν ο πίνακας Α των συντελεστών του γραμμικού συστήματος είναι αραιός πίνακας.
• Η πρώτη επαναληπτική μέθοδος είναι η επανάληψη Jacobi. Στην τεχνική αυτή, η i-οστή εξίσωση επιλύεται ως προς τον άγνωστο x_i και η εξίσωση αυτή χρησιμοποιείται για την ενημέρωση της τιμής του αγνώστου x_i. Συγκεκριμένα, η ενημέρωση γίνεται με τον ακόλουθο τύπο:

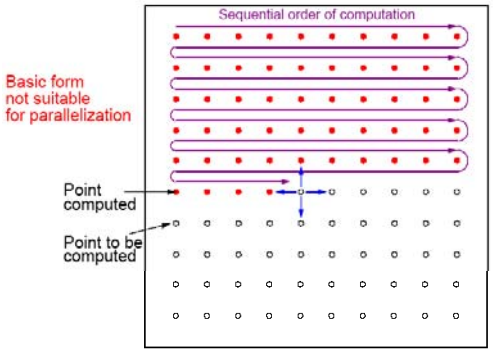
$$x_i^k = \frac{1}{a_{ii}}\left[b_i - \sum_{j \neq i} a_{ij} x_j^{k-1} \right]$$

όπου x_i^k είναι η τιμή του αγνώστου x_i μετά την κ-οστή επανάληψη, και x_i^{k-1} είναι η τιμή του αγνώστου x_j μετά την (κ-1)-οστή επανάληψη.
Έχει παρατηρηθεί ότι η σύγκλιση της τεχνικής Jacobi είναι σχετικά αργή και εξαρτάται από τον πίνακα συντελεστών Α.

• Η δεύτερη επαναληπτική τεχνική είναι η Gauss-Seidel τεχνική η οποία παρουσιάζει ταχύτερη σύγκλιση.
• Στη τεχνική αυτή, οι τιμές των αγνώστων που έχουν υπολογισθεί στη τρέχουσα επανάληψη, χρησιμοποιούνται για τον υπολογισμό των επόμενων αγνώστων. Συγκεκριμένα, η ενημέρωση της τιμής του αγνώστου x_i γίνεται με τον ακόλουθο τύπο:

$$x_i^k = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{N} a_{ij} x_j^{k-1} \right]$$

• Η επαναληπτική μέθοδος είναι εγγενώς ακολουθιακή αφού η ενημέρωση της τιμής του αγνώστου x_i εξαρτάται από τις νέες τιμές των αγνώστων x_j (j<i). Αντιθέτως, στην τεχνική Jacobi όλες οι ενημερώσεις των αγνώστων σε μία επανάληψη μπορούν να γίνουν παράλληλα αφού εξαρτώνται μόνο από τις τιμές των αγνώστων από την προηγούμενη επανάληψη.



Διάταξη Red-Black

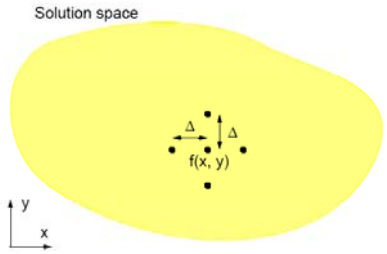
Πολλές φορές αραιά γραμμικά συστήματα εξισώσεων προκύπτουν ως διακριτή προσέγγιση μερικών διαφορικών εξισώσεων. Παράδειγμα, με τη «διακριτοποίηση» του επιπέδου η μερική διαφορική εξίσωση

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x^2} = 0$$

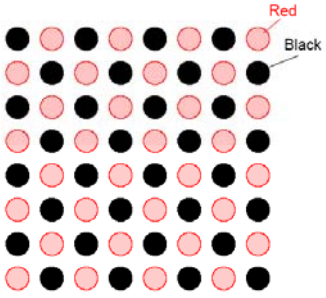
μπορεί να επιλυθεί προσεγγιστικά με τους ακόλουθους επαναληπτικούς τύπους

$$f_{i,j}^{k} = \frac{\left[f_{i-1,j}^{k-1} + f_{i,j-1}^{k-1} + f_{i+1,j}^{k-1} + f_{i,j+1}^{k-1} \right]}{4}$$

όπου $f_{i,j}$ είναι η τιμή της άγνωστης συνάρτησης στο σημείο (iΔx,jΔy).



Σε τέτοιου είδους γραμμικά συστήματα τα οποία έχουν προκύψει κατά την επίλυση μερικών διαφορικών εξισώσεων στο επίπεδο, υπάρχει ένας διαφορετικός τρόπος επίσκεψης των αγνώστων ο οποίος επιτρέπει τη παράλληλη εκτέλεση περισσότερων ενημερώσεων των τιμών των αγνώστων σε αντίθεση με την Gauss-Seidel μέθοδο.
Συγκεκριμένα οι άγνωστοι-σημεία στο επίπεδο χωρίζονται σε δύο κατηγορίες: τα κόκκινα και τα μαύρα σημεία. Όλες, οι ενημερώσεις στα σημεία του ίδιου χρώματος μπορεί να γίνουν παράλληλα αφού βασίζονται αποκλειστικά στις τιμές των σημείων του άλλου χρώματος.
Έτσι, ο αλγόριθμος εναλλάσσει δύο φάσεις όπου σε κάθε φάση ενημερώνει τις τιμές στα σημεία του ίδιου χρώματος.




```
forall (i = 1; i < n; i++)
        forall (j = 1; j < n; j++)
          if ((i + j) % 2 == 0)                    /* compute red points */
             f[i][j] = 0.25*(f[i-1][j] + f[i][j-1] + f[i+1][j] + f[i][j+1]);
   forall (i = 1; i < n; i++)
      forall (j = 1; j < n; j++)
        if ((i + j) % 2 != 0)                    /* compute black points */
           f[i][j] = 0.25*(f[i-1][j] + f[i][j-1] + f[i+1][j] + f[i][j+1]);
```