ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Επιστημονικός Υπολογισμός Ι

Ενότητα **3** : Βασικές Πράξεις Αριθμητικής Γραμμικής Άλγεβρας

Ευστράτιος Γαλλόπουλος
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο **«Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών»** έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σκοπός Ενότητας

- Θεμελιώδη προβλήματα της αριθμητικής γραμμικής άλγεβρας.
- Η ιεραρχία BLAS.
- Πλοκαδοποίηση και βασικές πράξεις στο υπολογιστικό μοντέλο ιεραρχικής μνήμης.
- Υπερταχύς πολλαπλασιασμός Strassen και παραλλαγές.

Περιεχόμενα

Από πρόσφατη δημοσίευση (FGHZ14)

Table 1
The increase of the computational complexity of an air pollution model in the period 1980–2014 (due to the refinement of the grids and to the involvement of more chemical species in the mathematical description of the underlying processes).

No.	Dimensionality of the model	Number of grid-points	Number of chemical species	Number of equations
1	2-D	32 × 32	2	2 048
2	2-D	32 × 32	35	35 840
3	3-D	32 × 32 × 10	35	358 400
4	2-D	96 × 96	35	322 560
5	3-D	96 × 96 × 10	35	3 225 600
6	2-D	480 × 480	35	8 064 000
7	3-D	480 × 480 × 10	35	80 640 000
8	2-D	480 × 480	56	12 902 400
9	3-D	480 × 480 × 10	56	139 024 000
10	2-D	480 × 480	169	38 937 600
11	3-D	480 × 480 × 10	169	389 376 000

Από άρθρο του A. Jaffe (Harvard), ``Ordering the Universe: The role of Mathematics'' (Jaf84)

Computers suffer from two fundamental limitations. Although the fastest computers can execute millions of operations in one second, they are always too slow. This may seem like a paradox, but the heart of the matter is: the bigger and better computers become, the larger are the problems scientists and engineers want to solve.

Υπενθύμιση και πρόγραμμα διάλεξης

- Στην προηγούμενη διάλεξη μιλήσαμε για τις πράξεις BLAS και εξετάσαμε ορισμένες από αυτές:

  BLAS-1 `DOT`, `_AXPY`

  BLAS-2 `GER`

  Αυτό έγινε στα πλαισια του υπολογιστικού μοντέλου μας.

- Στην προσπάθεια να μειώσουμε Φ_{min} όταν έχουμε περιορισμένη ΚΜ, είδαμε ότι για την `GER`, είναι απαραίτητο να κάνουμε πλοκαδοποίηση.

- Σήμερα εξετάζουμε την περίπτωση της BLAS-2 πράξης `MV`. Στη συνέχεια θα παρουσιάσουμε την πράξη που προσφερει τις περισσότερες δυνατότητες βελτίωσης της επίδοσης για το υπολογιστικό μοντέλο. Αυτό θα γίνει πάλι με πλοκαδοποίση και κατάλληλη οργάνωση των επιμέρους υπολογισμών.

Μητρώο $\times$ διάνυσμα: MV ή GAXPY

$$c \leftarrow c + Ab, c \in \mathbb{R}^{n_1 \times 1}, A \in \mathbb{R}^{n_1 \times n_3}, b \in \mathbb{R}^{n_3 \times 1}$$

ή

$$c \leftarrow c + a^\top B, c \in \mathbb{R}^{1 \times n_2}, a \in \mathbb{R}^{n_3 \times 1}, B \in \mathbb{R}^{n_3 \times n_2}$$

Εξετάζουμε την πρώτη περίπτωση.

- MV = A x plus y γι' αυτό και η εναλλακτική ορολογία `GAXPY`.
- $\Omega = 2n_1 n_3$, $\Phi_{\text{min}} = n_1 n_3 + 2n_1 + n_3$,
- $\mu_{\text{min}} = \frac{1}{2} + \frac{1}{2n_1} + \frac{1}{n_3}$.

Υλοποίηση

Μορφή DOT/ *ik*/ κ. γραμμές: τα στοιχεία του y υπολογίζονται από το εσωτερικό γινόμενο των γραμμών του A με το x,

$$\eta_i \leftarrow \eta_i + a_{i,:}^{\top} x = \eta_i + \sum_{k=1}^{n_3} \alpha_{ik} \xi_k, \quad i = 1, \ldots, n_1.$$

Μορφή sAXPY/ *ki* / κ. στήλες τα στοιχεία του y υπολογίζονται από τον γραμμικό συνδυασμό των στηλών του A:

$$y \leftarrow y + \sum_{k=1}^{n_3} a_{:,k} \xi_k.$$

Κώδικας 1: MV κατά γραμμές

```
1 for i=1:n1
2     for k = 1:n3
3         y(i) = y(i) + A(i,k)*x(k);
4     end
5 end
```

Κώδικας 2: MV κατά στήλες

```
1 for k=1:n3
2     for i = 1:n1
3         y(i) = y(i) + A(i,k)*x(k);
4     end
5 end
```

Υλοποιήσεις

Κώδικας 3: MV κατά γραμμές

```
1  LOAD(y,A,x);
2  for i=1:n1
3      for k = 1:n3
4          y(i) = y(i) + A(i,k)*x(k);
5      end
6  end
7  STORE(y)
```

- $\Phi = \Phi_{min} = 2n_1 + n_1 n_3 + n_3$.
- Απαιτείται $\mathcal{K} = O(n_1 n_3)$

Υλοποιήσεις

Κώδικας 4: MV κατά γραμμές

```
1  for i=1:n1
2      LOAD(y(i))
3      for k = 1:n3
4          LOAD(A(i,k),x(k))
5          y(i) = y(i) + A(i,k)*x(k);
6      end
7      STORE(y(i))
8  end
```

- $\Phi = \sum_{i=1}^{n_1}(2 + \sum_{j=1}^{n_3} 2) = 2n_1 + 2n_1 n_3$
- Απαιτείται $\mathcal{K} = O(1)$

Υλοποίηση αν $\mathcal{K} = O(n_3)$

Κώδικας 5: MV κατά γραμμές

```
1  LOAD(x)
2  for i=1:n1
3      LOAD(y(i))
4      for k = 1:n3
5          LOAD(A(i,k))
6          y(i) = y(i) + A(i,k)*x(k);
7      end
8      STORE(y(i))
9  end
```

- $\Phi = n_3 + \sum_{i=1}^{n_1}(2 + \sum_{j=1}^{n_3} 1) = n_3 + 2n_1 + n_1 n_3 = \Phi_{\text{min}}$
- Απαιτείται $\mathcal{K} = O(n_3)$
- Υλοποίηση _AXPY αν $\mathcal{K} = O(n_1)$
- Υβριδική υλοποίηση για $\mathcal{K} = O(\min(n_1, n_3))$

Γενική υλοποίηση

Έστω $n_3 = m_3 k_3$ και

$$A = [A_1, \cdots, A_{k_3}], x = [x_1; \cdots; x_{k_3}]$$

Διάσπαση σε υπογινόμενα:

$$y = y + A_1 x_1 + A_2 x_2 + \cdots A_{k_3} x_{k_3}$$

Αν $\mathcal{K} = O(m_3)$ τότε κάθε μερικό `MV/GAXPY` γίνεται με βέλτιστο $\Phi(J) = 2n_1 + n_1 m_3 + m_3$ άρα

$$\begin{aligned} \Phi &= \sum_{J=1}^{k_3} \Phi(J) \\ &= 2n_1 k_3 + n_1 n_3 + n_3. \end{aligned}$$

Εναλλακτική υλοποίηση

Έστω $n_1 = m_1 k_1$ και

$$y = [y_1; \cdots , ; y_{k_1}], A = [A_1; \cdots , ; A_{k_1}]$$

Διάσπαση:

$$y_I = y_I + A_I x, \quad I = 1 : k_1$$

Αν $\mathcal{K} = O(m_1)$ τότε κάθε μερικό `MV/GAXPY` γίνεται με βέλτιστο $\Phi(J) = 2m_1 + m_1 n_3 + n_3$ άρα

$$\begin{aligned} \Phi &= \sum_{J=1}^{k_1} \Phi(J) \\ &= 2n_1 + n_1 n_3 + k_1 n_3. \end{aligned}$$

Συμπέρασμα

- Η επιλογή εξαρτάται από τη σχέση n_1, n_3 με $\mathcal{K}$.
- Αν $n_1 = n_3 = n$ οι τιμές είναι $n^2 + 2n + k_1 n$ και $n^2 + 2k_3 n + n$

Για δεδομένο $\mathcal{K}$ και εφόσον $n_1 = n_3 = n$ ισχύει ότι $k_1 = k_3$ επομένως ο τεμαχισμός προς την n_3 απαιτεί λιγότερες μεταφορές.

Έχουμε

$$\begin{aligned} \Phi &= \min\left(\frac{2n_1 n_3}{m_3} + n_1 n_3 + n_3, 2n_1 + n_1 n_3 + \frac{n_1}{m_1} n_3\right) \\ &= \min\left(\frac{2n_1 n_3}{\mathcal{K}} + n_1 n_3 + n_3, 2n_1 + n_1 n_3 + \frac{n_1}{\mathcal{K}} n_3\right) \end{aligned}$$

Παρατήρηση Η καλύτερη υλοποίηση εξαρτάται από τη σχέση των διαστάσεων με το $\mathcal{K}$ και η επιλογή απαιτεί να εξετάσουμε το Φ για τις τιμές των δεικτών.

Κοινά χαρακτηριστικά MV, GER

- $n_i, n_j > 1, n_k = 1$ για $i, j, k \in \{1, 2, 3\}$
- $\Omega = O(n^2)$, $\Phi_{\text{min}} = O(n^2)$, $\mu_{\text{min}} = O(1)$
- Πολυπλοκότητες τετραγωνικές στην κυρίαρχη διάσταση
- Μη αποδοτική υλοποίηση σε ιεραρχική μνήμη
- Καλύτερο μ_{min} από τις DOT, _AXPY.

Βασικές πράξεις γραμμικής άλγεβρας 2ου επιπέδου

Παρατηρήσεις

Ο τεμαχισμός σε πλοκάδες (πλοκαδοποίηση - blocking) είναι κλειδί για την συγγραφή αποδοτικού κώδικα για ιεραρχική μνήμη.

- Η συνηθισμένες στρατηγικές λαμβάνουν υπόψη και παραμετροποιούνται βάσει των πόρων και ειδικότερα εν γνώσει των χαρακτηριστικών της κρυφής μνήμης (cache aware)[1]. όσο περισσότερες πληροφορίες για τους πόρους
- Για την εν λόγω στρατηγική, ο καλύτερος τεμαχισμός εξαρτάται από το μέγεθος της κρυφής μνήμης και των καταχωρητών.
- Η υλοποίηση που επιλέξαμε βασίζεται σε βέλτιστη υλοποίηση όμοιου αλλά μικρότερου προβλήματος με διαστάσεις που επιλέξαμε βάσει του μεγέθους της κρυφής μνήμης.
- Για να διαλέξουμε την καλύτερη υλοποίηση πρέπει να εξετάσουμε και εναλλακτικές εμφωλεύσεις των βρόχων!

[1] Είναι φυσιολογικό να αναμένουμε ότι όσο καλύτερη ενημέρωση υπάρχει για τα χαρακτηριστικά των πόρων του συστήματος, τόσο πιο βέλτιστο κώδικα μπορούμε να γράψουμε. Υπάρχουν μεθοδολογίες που προσπαθούν να πλησιάσουν βέλτιστη επίδοση εν αγνοία της κρυφής μνήμης και των άλλων πόρων (cache oblivious, resource oblivious).

Πολλαπλασιασμός γενικών μητρώων (πυκνών + αδόμητων)

$$C = C + AB, \quad C \in \mathbf{R}^{n_1 \times n_2}, A \in \mathbf{R}^{n_1 \times n_3}, B \in \mathbf{R}^{n_3 \times n_2}.$$

Κώδικας 6: Pseudo-MATLAB BLAS-3 block with L/S

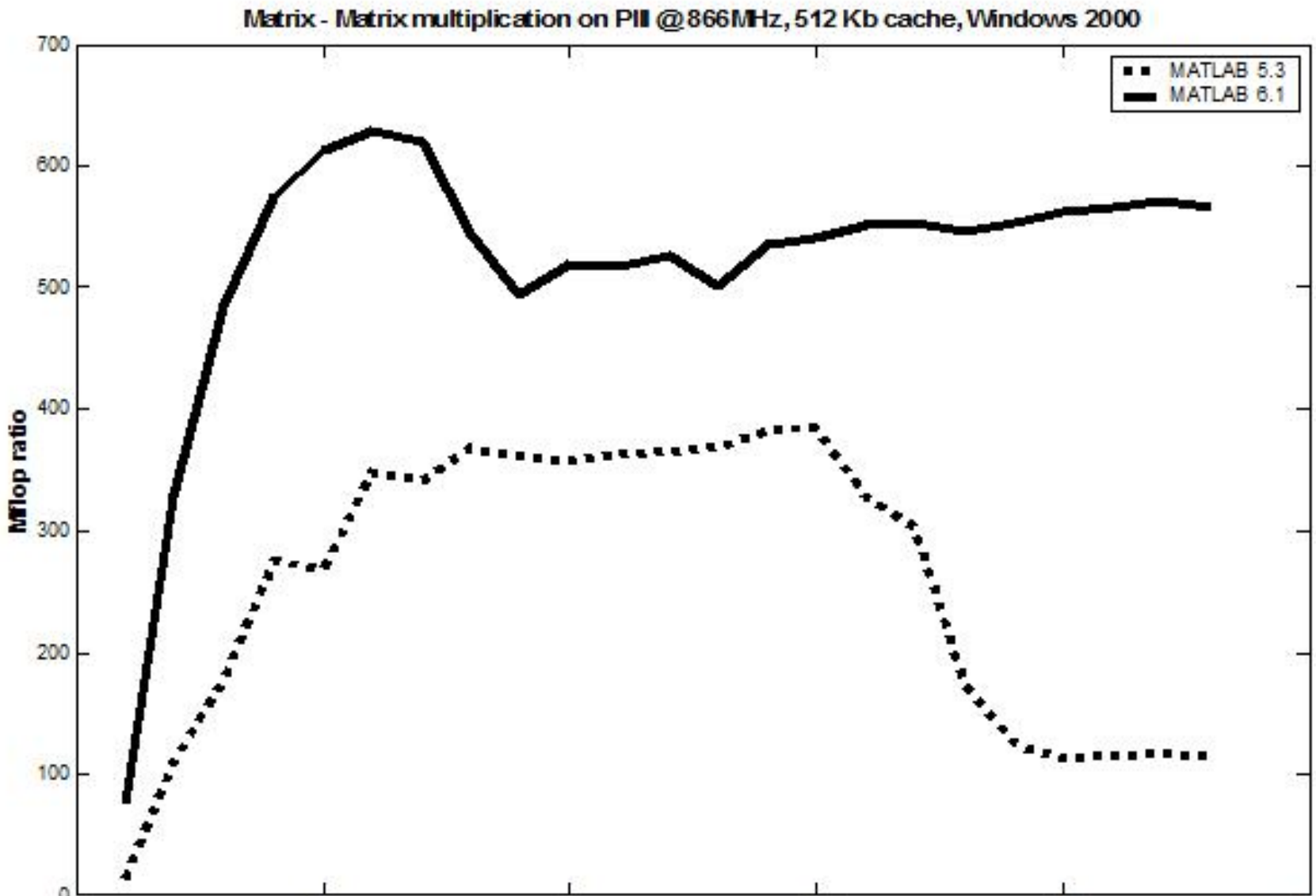
```
1  function [C] = mulmm(C,A,B)
2  LOAD (C,A,B)
3  for ?=1:n?
4      for ?=1:n?
5          for ?=1:n?
6              C(i,j) = C(i,j)+A(i,k)*B(k,j);
7          end
8      end
9  end
10 STORE(C)
```

$$\begin{aligned} [\mathcal{K}, \Omega, \Phi_{\text{min}}] &= [O(n_1 n_2 + n_1 n_3 + n_2 n_3), 2n_1 n_2 n_3, 2n_1 n_2 + (n_1 + n_2) n_3] \\ \Rightarrow \mu_{\text{min}} &= \frac{1}{n_3} + \frac{1}{2n_1} + \frac{1}{2n_2}. \end{aligned}$$

Γιατί μας ενδιαφέρει;

Πολλές εφαρμογές περιέχουν (μερικές φορές σε λανθάνουσα μορφή) πράξεις τύπου BLAS-3

Γιατί μας ενδιαφέρει;



Γιατί μας ενδιαφέρει;

Technical Articles and Newsletters

Overview | Search Technical Articles | Newsletters | Cleve's Corner Collection | Sign Up

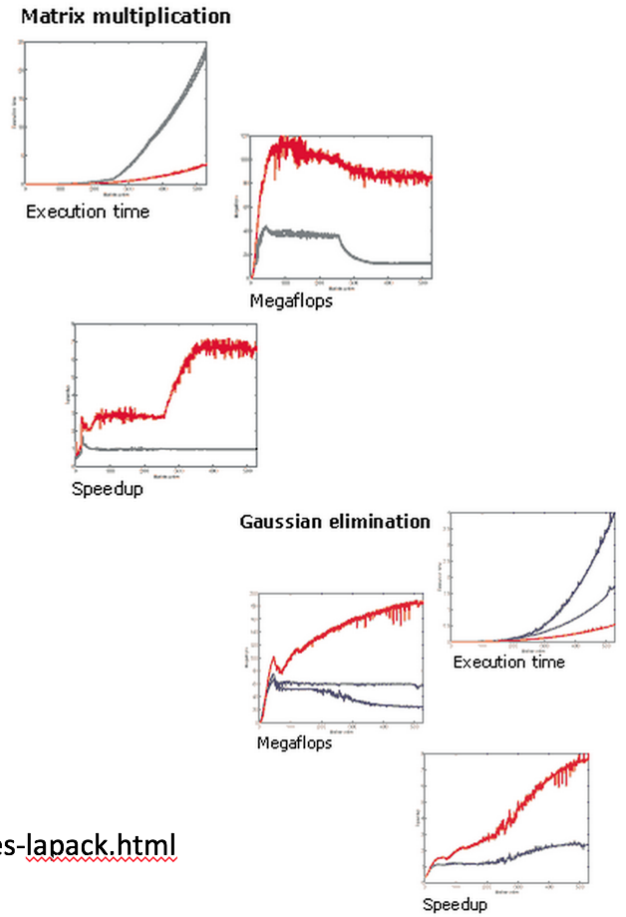
MATLAB Incorporates LAPACK

By Cleve Moler, MathWorks

MATLAB started its life in the late 1970s as an interactive calculator built on top of LINPACK and EISPACK, which were then state-of-the-art Fortran subroutine libraries for matrix computation. The mathematical core for all versions of MATLAB, up to version 5.3, has used translations to C of about a dozen of the Fortran subroutines from LINPACK and EISPACK.

LAPACK is the modern replacement for LINPACK and EISPACK. It is a large, multi-author, Fortran library for numerical linear algebra. A new version was released in July and is available from NETLIB (www.netlib.org/lapack). LAPACK was originally intended for use on supercomputers and other high-end machines. It uses block algorithms, which operate on several columns of a matrix at a time. On machines with high-speed cache memory, these block operations can provide a significant speed advantage. LAPACK also provides a more extensive set of capabilities than its predecessors do.

The speed of all these packages is closely related to the speed of the Basic Linear Algebra Subroutines, or BLAS. EISPACK did not use any BLAS. LINPACK used Level 1 BLAS, which operate on only one or two vectors, or columns of a matrix, at a time. Until now, MATLAB has used carefully coded C and assembly language versions of these Level 1 BLAS. LAPACK's block algorithms also make use of Level 2 and Level 3 BLAS, which operate on larger portions of entire matrices. The NETLIB distribution of LAPACK includes Reference BLAS written in Fortran. But the authors intended that various hardware and operating system manufacturers provide highly optimized, machine-specific, versions of the BLAS for their systems.



Δημοσιεύτηκε το 2000 στο Newsletter της Mathworks
http://www.mathworks.com/company/newsletters/articles/matlab-incorporates-lapack.html

ΠΡΟΣΟΧΗ

Αν τα n_j είναι ίσα με n τότε η ελάχιστη τιμή

$$\mu_{\min} = \mathcal{O}(\frac{1}{n})$$

είναι πολύ μικρότερη από τις τιμές που συναντήσαμε ως τώρα!

Ο πολλαπλασιασμός μητρώων φαίνεται να προσφέρει την δυνάμει μεγαλύτερη τοπικότητα μεταξύ των αλγορίθμων που συναντήσαμε ως τώρα! (Θυμηθείτε τα Mflop/s). Το ερώτημα είναι «πώς μπορούμε να τον υλοποιήσουμε για να τον αξιοποιήσουμε;»

Πολλαπλασιασμός μητρώων

Κώδικας 7: MATLAB BLAS-3

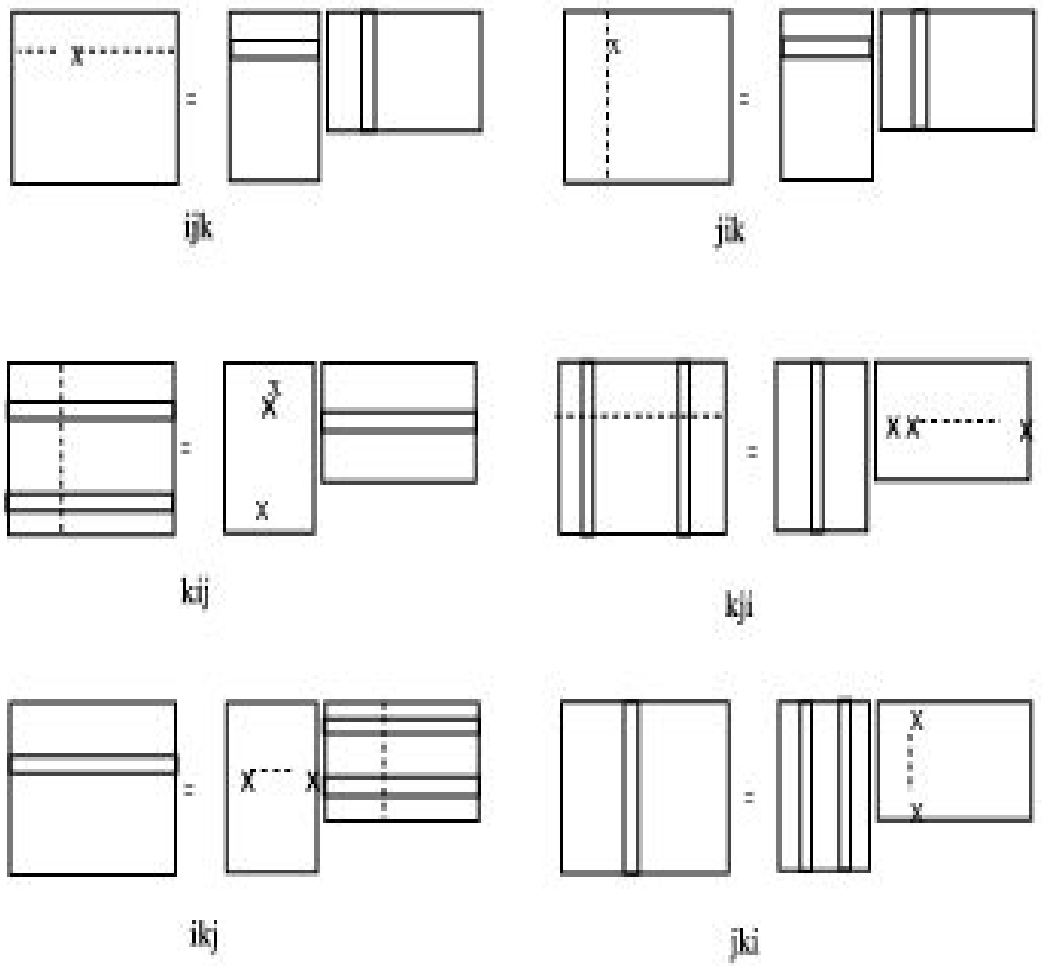
```
1  LOAD (C,A,B)
2  for ?=1:n?
3      for ?=1:n?
4          for ?=1:n?
5              C(i,j) = C(i,j)+A(i,k)*B(k,j);
6          end
7      end
8  end
9  STORE(C)
```

Παρατήρηση: Υπάρχουν 6 (= 3!) διαφορετικές δυνατές δυνατότητες εμφώλευσης. Συμβολίζονται συνήθως με την τριπλέτα των δεικτών γραμμένη με τη σειρά που γίνεται η εμφώλευση: i δεικτοδοτεί τις γραμμές των C, A, j τις στήλες των C, A και k τις στήλες του A και γραμμές του B.

Ερμηνεία εμφωλεύσεων (GL96)

σειρά βρόχου	**εσωτερικός βρόχος**	**μεσαίος βρόχος**	**τρόπος προσπέλασης**
ijk	DOT	MV/GAXPY βάσει DOT	*A* κ. στήλ., *B* κ. γραμμ.
jik	DOT	MV/GAXPY βάσει DOT	*B* κ. στήλ., *A* κ. γραμμ.
ikj	_AXPY	MV/GAXPY βάσει GAXPY	*B* κ. γραμμές
jki	_AXPY	MV/GAXPY βάσει GAXPY	*A* κ. στήλες
kij	_AXPY	GER	*B* κ. γραμμές
kji	_AXPY	GER	*A* κ. στήλες

Σχηματική αναπαράσταση



Παράδειγμα: πολλαπλασιασμός μητρώων με εμφώλευση *ijk*

Κώδικας 8: MATLAB BLAS-3

```
1  for i=1:n1
2      for j=1:n2
3          s = 0;
4          for k=1:n3
5              s = s+A(i,k)*B(k,j);
6          end
7          C(i,j)=s;
8      end
9  end
```

Σύσταση: Δοκιμάστε να ενθέσετε `LOAD, STORE` και εξετάστε τα προκύπτοντα $\mathcal{K}, \Phi_{\text{min}}$.

- Στο μοντέλο μας (και σε συστήματα με ιεραρχική μνήμη) επιλέγουμε υλοποιήσεις που έχουν για βάση πολλαπλασιασμούς υπομητρώων/πλοκάδων.
- Οδηγούμαστε έτσι σε αλγορίθμους με τριπλά εμφωλευμένο βρόχο μόνο που η εσωτερική έκφραση είναι και αυτή πράξη ανανέωσης υπομητρώου.

Πολλαπλασιασμός μητρώων σε πλοκάδες

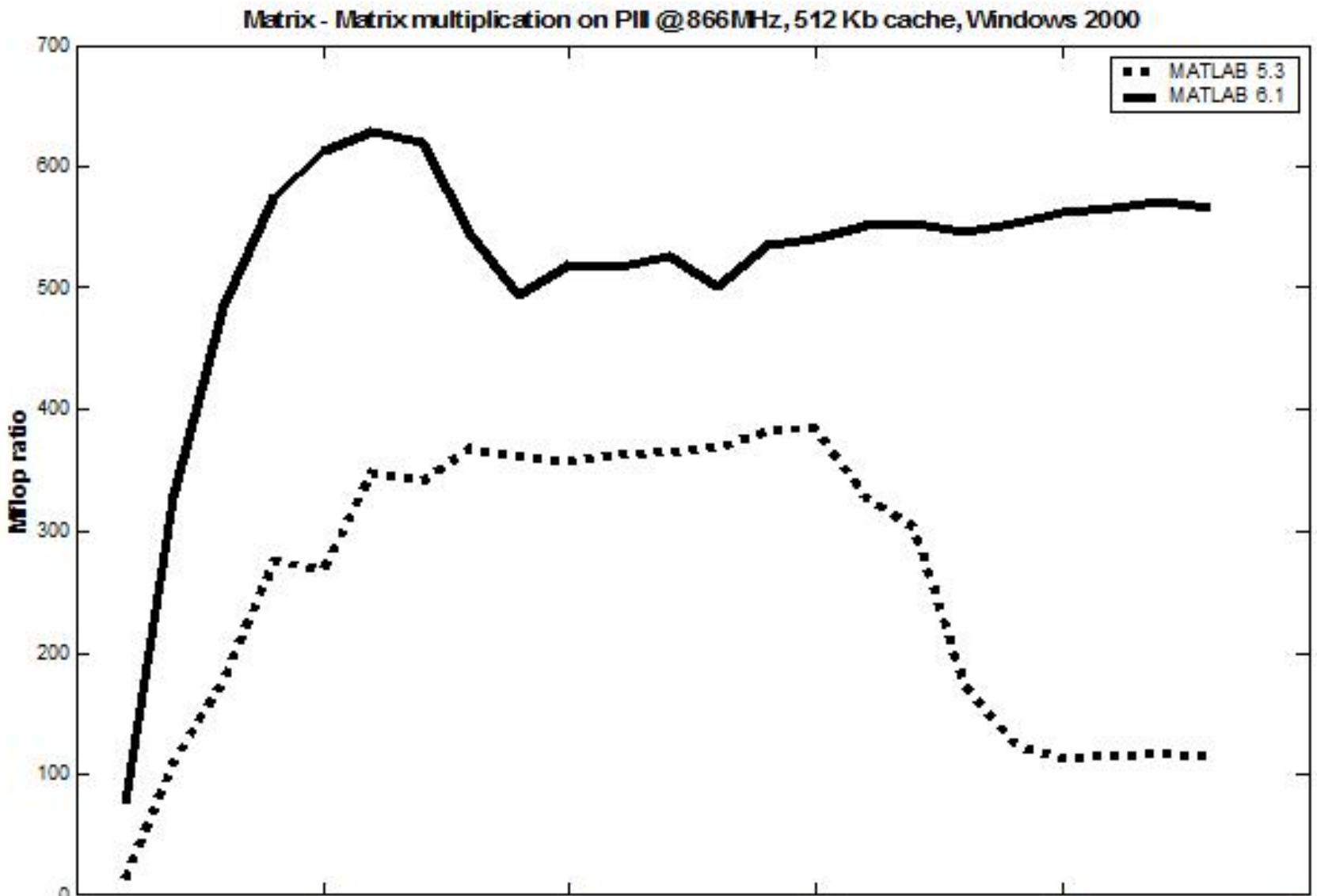
Κώδικας 9: MATLAB BLAS-3: blocked MM with IKJ outer

```
% !TeX encoding = ISO-8859-7
% 'eqei 'hdh g'inei arqikopo'ihsh
for I=1:k1
 for K=1:k3
  LOAD(A((I-1)*m1+1:I*m1,(K-1)*m3+1:K*m3))
  for J=1:k2
              LOAD(C((I-1)*m1+1:I*m1,(J-1)*m2+1:J*m2),...
                  B((K-1)*m3+1:K*m3,(J-1)*m2+1:J*m2))
      C((I-1)*m1+1:I*m1,(J-1)*m2+1:J*m2) = ...
                  C((I-1)*m1+1:I*m1,(J-1)*m2+1:J*m2) +...
                      A((I-1)*m1+1:I*m1,(K-1)*m3+1:K*m3)*...
                          B((K-1)*m3+1:K*m3,(J-1)*m2+1:J*m2)
               STORE(C((I-1)*m1+1:I*m1,(J-1)*m2+1:J*m2))
    end
  end
end
```

Ανάλυση

- Τα m_1, m_2, m_3 επελέγησαν έτσι ώστε ο «μικρός» πολλαπλασιασμός μητρώων (ΜΜ) $Z \leftarrow Z + XY$ όπου $Z \in \mathbb{R}^{m_1 \times m_2}, X \in \mathbb{R}^{m_1 \times m_3}, Y \in \mathbb{R}^{m_3 \times m_2}$ να εκτελείται με βέλτιστο αριθμό μεταφορών $\Phi_{\text{min}} = 2m_1m_2 + m_1m_3 + m_2m_3$.
- $$\Phi = n_1n_3 + n_1n_2n_3(\frac{1}{m_1} + \frac{2}{m_3}),$$
- Αν $\beta = m_1 = m_2 = m_3$ θα επιλέγαμε β τέτοιο ώστε $3\beta^2 \leq \mathcal{K}$.
- Σημ.: Προσεκτικότερη ανάλυση (όχι εδώ) δείχνει ότι το $\mathcal{K}$ μπορεί να μειωθεί σε $m_1m_3 + m_3$.

Μερικές οπτικοποιήσεις ≈ 2005 (hors-d'oeuvre)



Αποθήκευση πινάκων, προσπέλαση μνήμης και τοπικότητα

Σημαντικό θέμα: Η αποθήκευση πολυδιάστατων δεδομένων στη (γραμμική) μνήμη σε σχέση με τον τρόπο προσπέλασης στα δεδομένα από το πρόγραμμα και την τοπικότητα.

- Συνήθως η προσπέλαση στα στοιχεία πινάκων γίνεται με σταθερό βήμα (stride)
- Η μνήμη είναι μονοδιάστατη επομένως η συνήθης αποθήκευση 1-διάστατων πινάκων είναι απλή.
- ... οι αστοχίες μπορούν να μειωθούν αν η προσπέλαση γίνεται με βήμα μικρότερο του μεγέθους του «πλαισίου κρυφής μνήμης» cache line.
- Για πολυδιάστατους πίνακες, το θέμα είναι πιο περίπλοκο.
- Πρώτα πρέπει να ξέρουμε πώς αποθηκεύεται ο πίνακας.

Αποθήκευση γενικών πυκνών μητρώων

Στην παρούσα συζήτηση ενδιαφερόμαστε για τα πυκνά γενικά μητρώα: Δηλαδή, δεν υπάρχει καμία a priori γνώση σε σχέση με τη δομή τους που θα μπορούσε να εξειδικεύσει τον τρόπο αποθήκευσης.
Τι εννοούμε; Για παράδειγμα, αν γνωρίζαμε ότι το μητρώο είναι διαγώνιο, θα μπορούσαμε να αποθηκεύσουμε μόνον τη διαγώνιο ως ένα μονοδιάστατο πίνακα. Αν γνωρίζαμε ότι το μητρώο είναι συμμετρικό, θα μπορούσαμε να αποθηκεύσουμε μόνον τα μισά στοιχεία, κ.λπ.

Τρόποι αποθήκευσης γενικών μητρώων

Αν το $m \times n$ μητρώο αποθηκευτεί σε δομή δεδομένων που είναι $m \times n$ πίνακας και το πρώτο στοιχείο του πίνακα βρίσκεται στη μνήμη στη διεύθυνση $b_0 + 1$, διακρίνουμε 2 τρόπους απεικόνισης των στοιχείων του πίνακα στη μνήμη (προσοχή: Θεωρούμε ότι και στις 2 περιπτώσεις, οι τιμές των i, j αρχίζουν από 1 και όχι 0).

Διάταξη κατά γραμμές (row major order): (C, Pascal, Java). Το στοιχείο στη θέση (i, j) του πίνακα απεικονίζεται στη διεύθυνση $b_0 + (i-1)n + j$ της μνήμης. Δηλ. τα στοιχεία κάθε γραμμής βρίσκονται σε διαδοχικές διευθύνσεις, ενώ οι γραμμές αποθηκεύονται με τη σειρά που βρίσκονται στον πίνακα.

Διάταξη κατά στήλες (column major order): (Fortran, MATLAB, Octave). Το στοιχείο στη θέση (i, j) του πίνακα απεικονίζεται στη διεύθυνση $b_0 + (j-1)m + i$. Δηλ. τα στοιχεία κάθε στήλης βρίσκονται σε διαδοχικές διευθύνσεις, ενώ οι στήλες αποθηκεύονται με τη σειρά που βρίσκονται στον πίνακα.

Δείτε εδώ, το άρθρο της Wikipedia και εδώ σχετικά με την αποθήκευση πινάκων στη MATLAB.

Γραμμική δεικτοδότηση

R2011b Documentation → MATLAB

View documentation for other releases

Learn more about MATLAB

Linear Indexing

You can refer to the elements of a MATLAB matrix with a single subscript, `A(k)`. MATLAB stores matrices and arrays not in the shape that they appear when displayed in the MATLAB Commanc Window, but as a single column of elements. This single column is composed of all of the columns from the matrix, each appended to the last.

So, matrix `A`

```
A = [2 6 9; 4 2 8; 3 5 1]
A =
     2     6     9
     4     2     8
     3     5     1
```

is actually stored in memory as the sequence

```
2, 4, 3, 6, 2, 5, 9, 8, 1
```

The element at row 3, column 2 of matrix `A` (value = 5) can also be identified as element 6 in the actual storage sequence. To access this element, you have a choice of using the standard `A(3,2)` syntax, or you can use `A(6)`, which is referred to as *linear indexing*.

If you supply more subscripts, MATLAB calculates an index into the storage column based on the dimensions you assigned to the array. For example, assume a two-dimensional array like `A` has size `[d1 d2]`, where `d1` is the number of rows in the array and `d2` is the number of columns. If you supply two subscripts `(i, j)` representing row-column indices, the offset is

```
(j-1) * d1 + i
```

Given the expression `A(3,2)`, MATLAB calculates the offset into `A`'s storage column as `(2-1) * 3 + 3`, or 6. Counting down six elements in the column accesses the value 5.

Προσπέλαση διαδοχικών στοιχείων

Από τη μέθοδο αποθήκευσης εξαρτάται η σχετική θέση των στοιχείων του πίνακα στη μνήμη. Συνήθως, η απόσταση στις διευθύνσεις των στοιχείων τα οποία ζητούνται είναι σταθερή και καλείται βήμα ή διασκελισμός πρόσβασης (access stride).

Ερώτηση: Για πίνακα μεγέθους $m \times n$, ποιός είναι ο διασκελισμός για την πρόσβαση διαδοχικών στοιχείων κατά μήκος κάθε στήλης ή γραμμής;
Αν η διεύθυνση του στοιχείου στη θέση (i, j) του πίνακα είναι M και η διάταξη είναι:

κ. γραμμές	α) τότε το στοιχείο στη θέση $(i+1, j)$ είναι στη διεύθυνση $M + n$. β) το στοιχείο στη θέση $(i, j+1)$ είναι στη διεύθυνση $M + 1$.
κ. στήλες	α) τότε το στοιχείο στη θέση $(i+1, j)$ είναι στη διεύθυνση $M + 1$. β) το στοιχείο στη θέση $(i, j+1)$ είναι στη διεύθυνση $M + m$.

Επιθυμητό: η προσπέλαση στη μνήμη να γίνεται με μικρό διασκελισμό ώστε να υπάρχει τοπικότητα και να αξιοποιούνται τα στοιχεία που προσκομίζονται εντός μιας cache line.

Παράδειγμα προσπέλασης στη MATLAB

Στη MATLAB η αποθήκευση γίνεται κατά στήλες.

Κώδικας 10: MV κατά γραμμές $\Rightarrow$ διασκελισμός n_1.

```
1  for i=1:n1
2      for k = 1:n3
3          y(i) = y(i) + A(i,k)*x(k);
4      end
5  end
```

Κώδικας 11: MV κατά στήλες $\Rightarrow$ διασκελισμός 1.

```
1  for k=1:n3
2      for i = 1:n1
3          y(i) = y(i) + A(i,k)*x(k);
4      end
5  end
```

BLAS: Basic Linear Algebra Subprograms

Από τη Wikipedia:

Basic Linear Algebra Subprograms (BLAS) is a de facto application programming interface standard for publishing libraries to perform basic linear algebra operations such as vector and matrix multiplication. They were first published in 1979, and are used to build larger packages such as LAPACK. Heavily used in high-performance computing, highly optimized implementations of the BLAS interface have been developed by hardware vendors ...

Τρία επίπεδα: Εν συντομία BLAS-1, BLAS-2, BLAS-3. Γενικά, αναφέρονται σε πράξεις μεταξύ αλγεβρικών αντικειμένων (μητρώων, διανυσμάτων) στα οποία μπορεί να υφίστανται 3 διαστάσεις συνολικά (οι βαθμωτοί δεν λαμβάνονται υπόψη)

- BLAS-1: 1 διάστ. $>$ 1, δηλ. πράξεις μεταξύ διανυσμάτων (π.χ. DOT, _AXPY).
- BLAS-2: 2 διαστ. $>$ 1, δηλ. πράξεις μεταξύ μητρώων, διανυσμάτων (π.χ. MV, ανανέωση τάξης-1)..
- BLAS-3: 3 διαστ. $>$ 1, δηλ. πράξεις μεταξύ μητρώων (π.χ. MM).

BLAS: Basic Linear Algebra Subprograms

- API για βασικές πράξεις της ΑΓΑ. Περιγράφονται η ονοματολογία, ο τρόπος κλήσης και οι πράξεις, όχι όμως η υλοποίηση.
- Συνοπτική παρουσίαση http://www.netlib.org/lapack/lug/node145.html
- Κώδικες αναφοράς http://netlib.org/blas/
- Χρησιμότητα Από το εγχειρίδιο της LAPACK: The LAPACK strategy for combining efficiency with portability is to construct the software as much as possible out of calls to the BLAS; the BLAS are used as building blocks. The efficiency of LAPACK software depends on the efficient implementation of the BLAS being provided by computer vendors (or others) for their machines. Thus the BLAS form a low-level interface between LAPACK software and different machine architectures.

MATLAB και BLAS

Από τις σελίδες της Mathworks.

New Vendor BLAS Used for Linear Algebra in MATLAB

MATLAB uses Basic Linear Algebra Subprograms (BLAS) for its vector inner product, matrix-vector product, matrix-matrix product, and triangular solvers in \. MATLAB also uses BLAS behind its core numerical linear algebra routines from Linear Algebra Package (LAPACK), which are used in functions like `chol`, `lu`, `qr`, and within the linear system solver \.

On some platforms, MATLAB continues to use ATLAS BLAS.

Starting in Release 14, MATLAB 7.0 uses vendor BLAS from the vecLib library on the Mac.

Starting in Release 14 with Service Pack 1, MATLAB 7.0.1 uses vendor BLAS from

- The Intel Math Kernel Library (MKL) Version 7.0 on Intel chips running both Windows and Linux. See the MATLAB 7.0 Release Notes for how to use the multi-threaded capabilities of MKL.
- The AMD Core Math Library (ACML) Version 2.0 library on AMD chips, native 64 bit application

Overriding the Default BLAS Library on Sun/Solaris Systems

MATLAB uses the Basic Linear Algebra Subroutines (BLAS) libraries to speed up matrix multiplication and LAPACK-based functions like `eig`, `svd`, and \ (`mldivide`). At start-up, MATLAB selects the BLAS library to use.

For Release 14 with Service Pack 1, MATLAB still uses the ATLAS BLAS libraries on the Sun Microsystems Solaris Operating System. However, you can switch the BLAS library that MATLAB uses to the Sun Performance Library (Sunperf) BLAS, provided by Sun Microsystems.

Δείτε το άρθρο του Cleve Moler.

MATLAB και BLAS

« Release Notes ▸ Version 7 (R14) MATLAB Software ▸ Mathematics, MATLAB Version 7 (R14) ▸ Overriding the Default BLAS Library on Intel/Windows Systems

The incomplete gamma function, `gammainc`, now accepts the input argument `tail`, using the syntax

```
Y = gammainc(X,A,tail)
```

`tail` specifies the tail of the incomplete gamma function when `X` is non-negative. The choices are for `tail` are `'lower'` (the default) and `'upper'`. The upper incomplete gamma function is defined as

```
1 - gammainc(x,a)
```

Overriding the Default BLAS Library on Intel/Windows Systems

Note Intel has used aggressive optimization to compile MKL. This optimization causes `NaNs` to be treated as zeros in some situations. Calculations that do not involve `NaNs` are done correctly. In some calculations that do involve `NaNs`, the `NaNs` will not propagate.

MATLAB uses the Basic Linear Algebra Subroutines (BLAS) libraries to speed up matrix multiplication and LAPACK-based functions like `eig`, `svd`, and \ (`mldivide`). At start-up, MATLAB selects the BLAS library to use.

For R14, MATLAB still uses the ATLAS BLAS libraries, however, on Windows systems running on Intel processors, you can switch the BLAS library that MATLAB uses to the Math Kernel Library (MKL) BLAS, provided by Intel.

If you want to take advantage of the potential performance enhancements provided by the Intel BLAS, you can set the value of the environment variable `BLAS_VERSION` to the name of the MKL library, `mkl.dll`. MATLAB uses the BLAS specified by this environment variable, if it exists.

To set the `BLAS_VERSION` environment variable, follow this procedure:

1. Click the **Start** button, go to the **Settings** menu, and select **Control Panel**.
2. On the **Control Panel** menu, select **System**.
3. In the **System Properties** dialog box, click the **Advanced** tab.
4. On the **Advanced** panel, click the **Environment Variables** button.
5. In the **Environment Variables** dialog box, click the **New** button in the User variables section.
6. In the **New User Variable** dialog box, enter the name of the variable as `BLAS_VERSION` and set the value of the variable to the name of the MKL library: `mkl.dll`.

Βιβλιογραφία I

I. Faragó, K. Georgiev, Á Havasi, and Z. Zlatev.

Efficient algorithms for large scale scientific computations: Introduction.

Computers and Mathematics with Applications, 67(12):2085--2087, 2014.

G.H. Golub and C.F. Van Loan.

Matrix Computations.

The Johns Hopkins University Press, Baltimore, 3d edition, 1996.

A. Jaffe.

Ordering the universe: The role of mathematics.

SIAM Rev., 26(4):473--Hong 500, 1984.

Ε. Γαλλόπουλος.

Επιστημονικός Υπολογισμός Ι.

Πανεπιστήμιο Πατρών, 2008.

Χρήση Έργου Τρίτων Ι

1. http://epubs.siam.org/doi/abs/10.1137/1026103 (βλ. σελ 5)
2. http://www.mathworks.com/company/newsletters/articles/matlab-incorporates-lapack.html (βλ. σελ 19)
3. http://www.mathworks.com/help/matlab/math/matrix-indexing.html#f1-85511 (βλ. σελ 14)
4. http://www.netlib.org/lapack/lug/node65.html (βλ. σελ 35)
5. http://www.mathworks.com/help/matlab/rn/f14-998197.html (βλ. σελ 36)
6. http://www.mathworks.com/company/newsletters/articles/matlab-incorporates-lapack.html (βλ. σελ 36)
7. MATLAB Release Notes (βλ. σελ 37)

Σημείωμα Αναφοράς

``Επιστημονικός Υπολογισμός Ι'', Έκδοση: 1.0, Πάτρα 2013-2014.

Διαθέσιμο από τη δικτυακή διεύθυνση: https://eclass.upatras.gr/courses/CEID1096/

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης