Πίνακες και διανύσματα - λογισμικό για χρήση τους

- Υπάρχουν πολλά προγράμματα που έχουν γραφεί για χρήση πινάκων, λύσεις συστημάτων εξισώσεων, ιδιοτιμές κλπ
- Freeware και μπορεί να το κάνετε install σε ubuntu/mac ακόμα και windows
  - LINPACK: πακέτο για γραμμικές εξισώσεις και προβλήματα ελαχίστων τετραγώνων
  - LAPACK: πακέτο για λύση συμμετρικών, ασυμμετρικών και γενικών προβλημάτων ιδιοτιμών. Πληροφορίες για το πακέτο στην http://www.netlib.org
    Μπορείτε να βρείτε τον κώδικα και περιγραφές των συναρτήσεων
  - BLAS (I, II, III) (Basic Linear Algebra Subprograms): subroutines που δίνουν τα βασικά εργαλεία για πράξεις με πίνακες και διανύσματα. BLAS I είναι για διανύσματα,
    BLAS II για πράξεις διανυσμάτων-πινάκων και BLAS III για πράξεις πινάκων–πινάκων.
    Διαθέσιμος κώδικας στην διεύθυνση: http://www.netlib.org
- Για τους χρήστες ubuntu χρειάζεται να κάνετε: (για χρήστες mac είναι ήδη εγκατεστημένο)

```
sudo apt-get install libblas3gf
sudo apt-get install libblas-dev
sudo apt-get install libblas-doc
sudo apt-get install liblapacke-dev
sudo apt-get install liblapack3gf
sudo apt-get install liblapack-dev
sudo apt-get install liblapack-doc
```

Compiling/Linking

- Για να κάνετε compile/link τον κώδικά σας θα πρέπει να δώσετε την εντολή:

```
g++ κώδικας.C –I "/usr/include" –L"/usr/lib" –llapacke –lblas  -o  executable name
```

- Ο κώδικάς σας θα πρέπει να έχει δυο σχετικά header files:

```
#include <cblas.h>
#include <lapacke.h>
```

- Για mac χρήστες:
  - Ο κώδικάς σας θα πρέπει να έχει τα header files:

```
#include <Accelerate/Accelerate.h>
#include <vecLib/clapack.h>
```

  - Για να κάνετε compile/link θα πρέπει να δώσετε την εντολή:

```
g++ κώδικας.C –framework Accelerate –o executable name
```

- Browser για τον κώδικα μπορεί να βρεθεί στο link:

  http://www.netlib.org/lapack/explore-html/index.html

Απλό παράδειγμα – ubuntu/mac

- Ένα απλό παράδειγμα: έστω θέλουμε την λύση του συστήματος $\begin{matrix} x+y=2 \\ x-y=0 \end{matrix}$ με λύση (x,y)=(1,1)

```
#include <iostream>
#include <vector>
#include <Accelerate/Accelerate.h>
using namespace std;
int main() {
    char trans = 'N';
    int dim = 2;     int nrhs = 1;
    int LDA = dim; int LDB = dim;
    int info;

    vector<double> a, b;

    a.push_back(1); a.push_back(1);
    a.push_back(1); a.push_back(-1);
    b.push_back(2); b.push_back(0);
    int ipiv[3];

    dgetrf_(&dim, &dim, &*a.begin(), &LDA, ipiv, &info);
    dgetrs_(&trans, &dim, &nrhs, & *a.begin(), &LDA, ipiv, & *b.begin(), &LDB, &info);
// You can use simply:
//  dgesv_(&dim, &nrhs, &*a.begin(), &LDA,ipiv, & *b.begin(), &LDB, &info);
    cout << "solution is:";
    cout << "[" << b[0] << ", " << b[1] << ", " << "]" << endl;
    cout << "Info = " << info << endl;
    return 0;
}
```

θα μπορούσαμε να γράψουμε: &a[0]

αν ορίζαμε πίνακες α[2][2] και b[0][2] θα περνούσαμε &a[0][0], και β[0][0]

Πίνακες – βασικά

- Ιδιότητες των πινάκων

$$\mathrm{A}=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \text{και} \quad \mathrm{I}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Ο αντίστροφος είναι: $\mathrm{A}^{-1}\mathrm{A}=\mathrm{I}$

- Βασικές Ιδιότητες

Σχέση	Όνομα	Στοιχεία
$\mathrm{A}=\mathrm{A}^{\mathrm{T}}$	συμμετρικός	$a_{ij}=a_{ji}$
$\mathrm{A}=\left(\mathrm{A}^{\mathrm{T}}\right)^{-1}$	πραγματικός ορθογώνιος	$\sum_k a_{ik}a_{jk}=\sum_k a_{ki}a_{kj}=\delta_{ij}$
$\mathrm{A}=\mathrm{A}^{*}$	πραγματικός πίνακας	$a_{ij}=a_{ij}^{*}$
$\mathrm{A}=\mathrm{A}^{\dagger}$	ερμιτιανός	$a_{ij}=a_{ji}^{*}$
$\mathrm{A}=\left(\mathrm{A}^{\dagger}\right)^{-1}$	μοναδιακός	$\sum_k a_{ik}a_{jk}^{*}=\sum_k a_{ki}a_{kj}^{*}=\delta_{ij}$

Μερικοί γνωστοί πίνακες

- ✓ Διαγώνιος αν $a_{ij} = 0$ για $i \neq j$
- ✓ Πάνω τριγωνικός αν $a_{ij} = 0$ για $i > j$
- ✓ Κάτω τριγωνικός αν $a_{ij} = 0$ για $i < j$
- ✓ Πάνω Hessenberg αν $a_{ij} = 0$ για $i > j+1$
- ✓ Κάτω Hessenberg αν $a_{ij} = 0$ για $i < j+1$
- ✓ Τριγωνικός αν $a_{ij} = 0$ για $|i-j| > 1$
- ✓ Κάτω ζωνικός αν $a_{ij} = 0$ για $i > j+p$ όπου p το εύρος της ζώνης
- ✓ Πάνω ζωνικός αν $a_{ij} = 0$ για $i < j+p$ όπου p το εύρος της ζώνης
- ✓ Ζωνικός, συγκεκριμένοι πάνω, κάτω τριγωνικοί

Βασικές ιδιότητες

- Μερικές ισοδύναμες σχέσεις/ορισμοί
- Για ένα NxN πίνακα A, τα ακόλουθα είναι ισοδύναμα:
  - ✓ Αν ο αντίστροφος του A υπάρχει, τότε ο A είναι αντιστρέψιμος
  - ✓ Η εξίσωση Ax=0 υπονοεί ότι x=0
  - ✓ Οι γραμμές του A σχηματίζουν μια βάση R^N
  - ✓ Οι στήλες του A σχηματίζουν μια βάση R^N
  - ✓ Ο A είναι γινόμενο στοιχειωδών πινάκων
  - ✓ Η τιμή 0 δεν είναι ιδιοτιμή του πίνακα A

Βασικές πράξεις με πίνακες/διανύσματα

- Οι κύριες πράξεις πινάκων/διανυσμάτων είναι η πρόσθεση και αφαίρεση

$$\mathrm{A} = \mathrm{B} \pm C \;\Rightarrow a_{ij} = b_{ij} \pm c_{ij}$$

- Βαθμωτός πολλαπλασιασμός:

$$\mathrm{A} = \gamma \mathrm{B} \;\Rightarrow a_{ij} = \gamma b_{ij}$$

- Πολλαπλασιασμός πίνακα με διάνυσμα:

$$y = \mathrm{A}x \;\Rightarrow y_i = \sum_{j=1}^{n} a_{ij} x_j$$

- Πολλαπλασιασμός πίνακα με πίνακα:

$$\mathrm{A} = \mathrm{B}C \Rightarrow a_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj}$$

- Αναστροφή:

$$\mathrm{A} = \mathrm{B}^{\mathrm{T}} \;\Rightarrow a_{ij} = b_{ji}$$

- Εσωτερικό γινόμενο διανυσμάτων: (αποτέλεσμα σταθερά)

$$x = y^T z \;\Rightarrow x = \sum_{j=1}^{n} y_j z_j$$

- Εξωτερικό γινόμενο διανυσμάτων: (αποτέλεσμα πίνακας)

$$\mathrm{A} = yz^T \;\Rightarrow a_{ij} = y_i z_j$$

Τρόπος χρήσης πινάκων στη C++

- **Στατικά**
  - Ορίζουμε ένα NxN πίνακα με N = 100

```
int N = 100;
double A[100][100];
for (i=0; i< N; i++){
   for (j=0; j< N; j++) {
       A[i][j] = 0;
   }
}
```

  - Ο πίνακας είναι οργανωμένος ως προς γραμμές
  - Για πρόσθεση/αφαίρεση θα γράφαμε:

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

  - Για πολλαπλασιασμό θα γράφαμε:

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

Τρόπος χρήσης πινάκων στη C++

- **Δυναμικά**

```
int N;
double ** A;
A = new double* [N];
for (i = 0;  i< N; i++){
    A[i] = new double [N];
}
```

Είναι καλό να ελευθερώνεται η μνήμη μετά την χρήση

```
for (i = 0;  i< N; i++){
    delete [] A[i];
}
delete [] A;
```