Διαδικαστικός Προγραμματισμός

(ECE_Y215)

Επιπρόσθετες Βιβλιοθήκες στη C

Στόχοι

- Ο σκοπός της σημερινής διάλεξης είναι η περαιτέρω παρουσίαση της GSL **με έμφαση στις διεπαφές της GSL με τη γραμμική άλγεβρα**
- Να αναφερθούμε σε **παραδείγματα** ώστε να εμπεδώσουμε καλύτερα τις θεωρητικές γνώσεις
- Να υποστηρίξει την περεταίρω ενασχόληση των φοιτητών στο αντικείμενο της διάλεξης μέσω της **2ης Δραστηριότητας**

- Η σημερινή διάλεξη συνοδεύεται **από τη 2η δραστηριότητα**
- Ημερομηνία παράδοσης: **28 Μαΐου 2021**
- +0.25 μονάδες στον τελικό βαθμό του μαθήματος

Από την προηγούμενη διάλεξη

Βιβλιοθήκη είναι μια συλλογή **μεταγλωττισμένων μονάδων** που μπορούν να συνδεθούν (**linked**) στα προγράμματά μας μέσω των διεπαφών (**header files - interfaces**) που παρέχουν.

Με άλλα λόγια κάθε βιβλιοθήκη αποτελείτε από **δυαδικά** αρχεία σε object code (*.o) που περιέχουν **υλοποιήσεις** όλων των συναρτήσεων που έχουν **δηλωθεί** στα αρχεία επικεφαλίδων .h

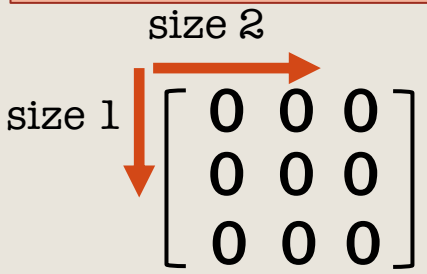

```
• # include <stdio.h>
• # include <gsl/gsl_matrix.h>
• …
```

Διεπαφή Πινάκων με τη GSL

Η διεπαφή για την πρόσβαση στις συναρτήσεις διαχείρισης πινάκων είναι: #include <gsl/gsl_matrix.h>, #include <gsl/gsl_vector.h>

```
typedef struct {
size_t size1;
size_t size2;
...
double * data;
} gsl_matrix;
```

```
gsl_matrix * A =
gsl_matrix_alloc (3, 3);
```



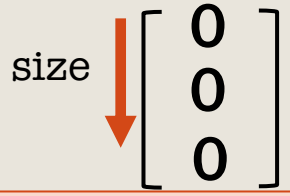

```
gsl_matrix_set_all
(A, 3.0);
```

...

$$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

```
typedef struct {
size_t size;
...
double * data
} gsl_vector;
```

```
gsl_vector *v=
gsl_vector_alloc(3);
```




```
gsl_vector_set_all
(A, 5.0);
```

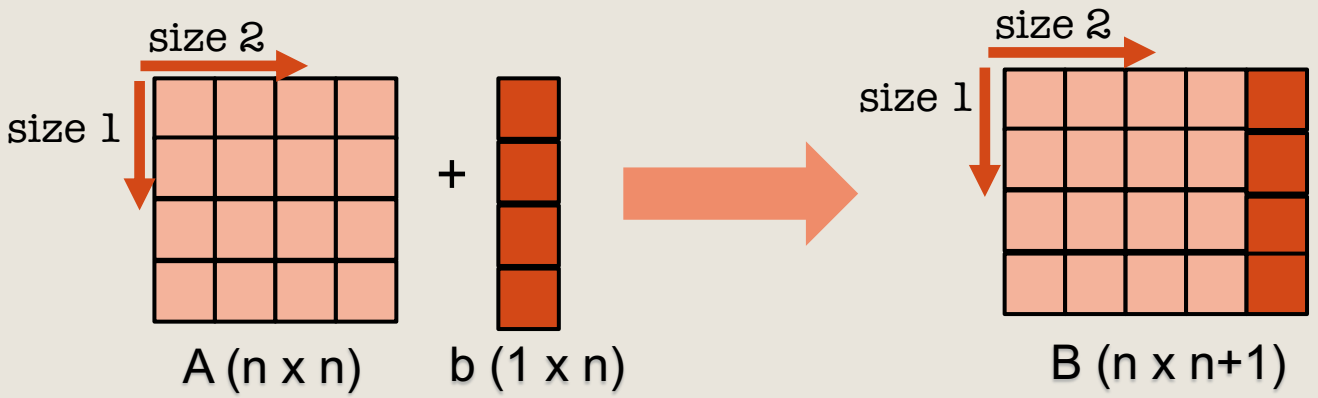
...

$$\begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}$$

https://www.gnu.org/software/gsl/doc/html/vectors.html#matrices
https://www.gnu.org/software/gsl/doc/html/vectors.html#vectors

Δημιουργία Επαυξημένου Πίνακα

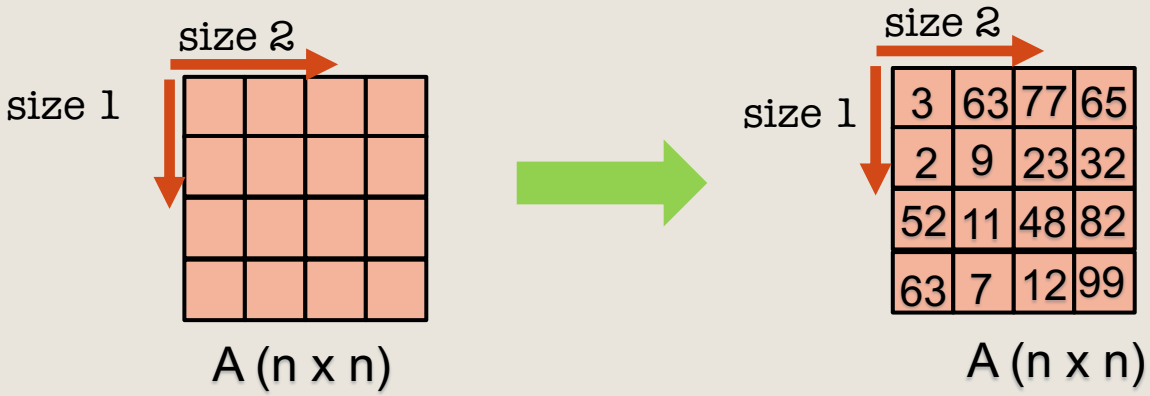
Διεπαφές: **#include <gsl/gsl_matrix.h>** και **#include <gsl/gsl_vector.h>**




```
for (i = 0; i < A->size1; i++){
      gsl_vector * tmp = gsl_vector_alloc (A->size1);
      gsl_matrix_get_col (tmp, A, i);
      gsl_matrix_set_col(B,i,tmp);
      gsl_vector_free (tmp);
}
gsl_matrix_set_col(B,B->size2-1,b);
```

Αρχικοποίηση Πίνακα με Γεννήτρια Τυχαίων Αριθμών

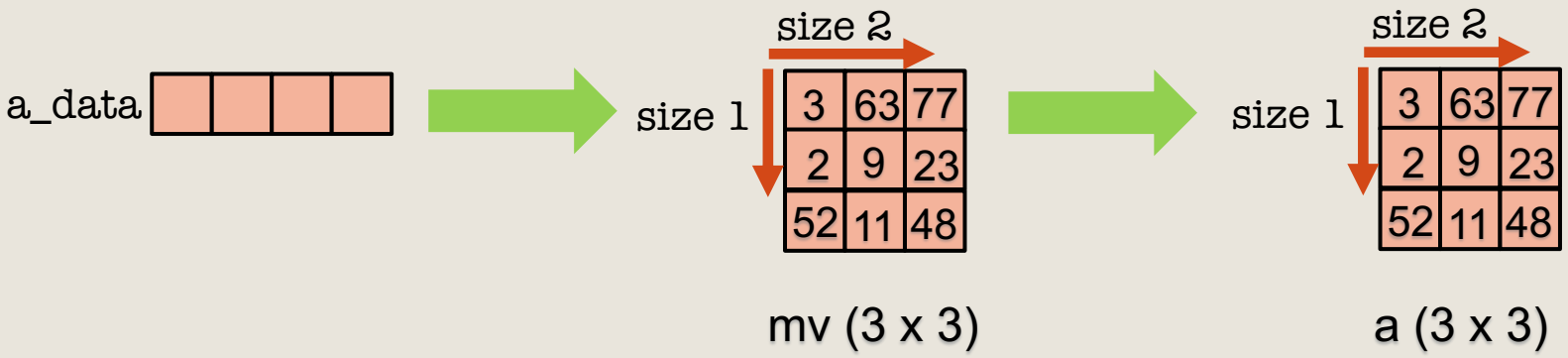
Διεπαφές: **#include <gsl/gsl_matrix.h>** και **#include <gsl/gsl_vector.h> και #include <gsl/gsl_rng.h>**




```
gsl_rng *mt_rng=seed_gsl_mt_rng();
int mt_res;
for (i = 0; i < a->size1; i++){
    for (j = 0; j < a->size2; j++) {
            mt_res = gsl_rng_uniform (mt_rng) * 999+1;
            gsl_matrix_set (a, i, j, mt_res);
     }
}
```

Αρχικοποίηση Πίνακα μέσω Όψεων Πινάκων

Δημιουργία πολλαπλών όψεων πινάκων μέσω της διεπαφής:
#include <gsl/gsl_matrix.h>




```
double  a_data[]={3,63,77,2,9,23,52,11,48};
gsl_matrix_view mv=gsl_matrix_view_array(a_data,3,3);
gsl_matrix_memcpy (a, &mv.matrix);

double b_data[]={12,16,8};
gsl_vector_view vv = gsl_vector_view_array (b_data, b->size);
gsl_vector_memcpy (b, &vv.vector);
```

Όψεις Πινάκων με τη GSL

Δημιουργία πολλαπλών όψεων πινάκων μέσω της διεπαφής:
#include <gsl/gsl_matrix.h>


```
gsl_matrix_view mv = gsl_matrix_submatrix(A, 0, 0, 2, 2);
```

Το mv είναι ένας δείκτης στην όψη του πίνακα.

Τριγωνοποίηση Επαυξημένου Πίνακα

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots \alpha_{1n} & b_{1'} \\ \alpha_{21} & \alpha_{22} & \dots \alpha_{2n} & b_{2'} \\ \vdots & & \dots \vdots & \dots \\ \alpha_{n1} & \alpha_{n2} & \dots \alpha_{nn} & b_{n'} \end{bmatrix}$$

Echelon form matrix

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots \alpha_{1n} & b_{1'} \\ 0 & \alpha_{22} & \dots \alpha_{2n} & b_{2'} \\ \vdots & & \dots \vdots & \dots \\ 0 & 0 & \dots \alpha_{nn} & b_{n'} \end{bmatrix}$$

Ο άγνωστος x_1 απαλείφεται από τη $2^η$,$3^η$,…n-εξίσωση, αφαιρώντας:

$m_{21} = a_{21}/a_{11}$ *φορές την 1η από τη 2η εξίσωση*

$m_{31} = a_{31}/a_{11}$ *φορές την 1η από τη 3η εξίσωση*

⋮

$m_{n1} = a_{n1}/a_{11}$ *φορές την 1η από τη n- εξίσωση*

Όπου $m_{21}, m_{31,\dots}, m_{n1}$ **πολλαπλασιαστές του Gauss για το πρώτο βήμα**

```
for(i=1;i<=n-1;i++){
        for(j=i+1;j<=n;j++){
           ratio = gsl_matrix_get (A, j, i)/gsl_matrix_get (A, i, i);
             ⋮
```

Προς τα πίσω αντικατάσταση

Echelon form matrix

$$\left[\begin{array}{cccc|c} \alpha_{11} & \alpha_{12} & \ldots \alpha_{1n} & & b_{1'} \\ 0 & \alpha_{22} & \ldots \alpha_{2n} & & b_{2'} \\ \vdots & & \ldots & \vdots & \ldots \\ 0 & 0 & \ldots. & \alpha_{nn} & b_{n'} \end{array}\right]$$

Προς τα πίσω αντικατάσταση της μεθόδου απαλοιφής του Gauss

$$X_n = b_{n'} / \alpha_{nn'}$$

$$X_{n-1} = (b_{n-1'} - X_n * \alpha_{n-1n}) / \alpha_{n-1',n-1}$$

$$\vdots$$

FLOP ως μονάδα μέτρησης της απαιτούμενης επεξεργαστικής ισχύς σε ένα υπολογιστικό πρόβλημα. Παράδειγμα:

```
ls (2,2)        flop for bsub :4
ls (3,3)        flop for bsub :9
ls (4,4)        flop for bsub :16
ls (5,5)        flop for bsub :25
ls (6,6)        flop for bsub :36
ls (7,7)        flop for bsub :49
ls (8,8)        flop for bsub :64
ls (9,9)        flop for bsub :81
ls (10,10)      flop for bsub :100
```

$$\text{FLOP}=1+\sum_{i=2}^{i=n}\ [(i-1)*2+1]$$

Πράξεις μεταξύ Διανυσμάτων & Πινάκων

Ο πηγαίος κώδικας του παραδείγματος βρίσκεται στο eclass:
GSL-BLAS-OPERATIONS

```
#include <stdio.h>
#include <stdlib.h>
#include <gsl/gsl_blas.h>

void gsl_my_print_matrix(gsl_matrix * m);
void gsl_my_print_vector(gsl_vector * v);
void gsl_print_matrix_view_elements(gsl_matrix_view mv);

void gsl_blas_level1_example();
void gsl_blas_level2_example();
void gsl_blas_level3_example();

int main()
{
    printf("\n *** BLAS level 1***");
    gsl_blas_level1_example();
    printf("\n *** BLAS level 2***");
    gsl_blas_level2_example();
    printf("\n *** BLAS level 3***");
    gsl_blas_level3_example();
    return 0;
}

void gsl_blas_level1_example(){
   float alpha;
```

$y = \alpha x + y$

$y = \alpha Ax + \beta y$

$C = \alpha AB + \beta C$

```
 *** BLAS LEVEL 3***
[A]
[0,0)] = 3      [0,1)] = 3      [0,2)] = 3
[1,0)] = 3      [1,1)] = 3      [1,2)] = 3
[2,0)] = 3      [2,1)] = 3      [2,2)] = 3

 [B]
[0,0)] = 3      [0,1)] = 3      [0,2)] = 3
[1,0)] = 3      [1,1)] = 3      [1,2)] = 3
[2,0)] = 3      [2,1)] = 3      [2,2)] = 3

 BLAS level 3: C = a AB +bC where (a=1.0 and b=0.0)
[0,0)] = 27     [0,1)] = 27     [0,2)] = 27
[1,0)] = 27     [1,1)] = 27     [1,2)] = 27
[2,0)] = 27     [2,1)] = 27     [2,2)] = 27
```

Ο πηγαίος κώδικας των παραδειγμάτων υπάρχει στο eclass [lecture 20]

Παράδειγμα με τη GSL- Πράξεις μεταξύ στοιχείων πίνακα

Ο πηγαίος κώδικας του παραδείγματος βρίσκεται στο eclass: **GSL-MATRIX-OPERATIONS**

```c
#include <stdio.h>
#include <stdlib.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
void gsl_my_print_matrix(gsl_matrix * m);
void gsl_my_transpose_matrix(gsl_matrix * ma);
void gsl_my_add_matrix_elements(gsl_matrix * ma,gsl_matrix * mb);
void gsl_my_sub_matrix_elements(gsl_matrix * ma,gsl_matrix * mb);
void gsl_my_mul_matrix_elements(gsl_matrix * ma,gsl_matrix * mb);
void gsl_my_div_matrix_elements(gsl_matrix * ma,gsl_matrix * mb);
void gsl_my_matrix_scale(gsl_matrix * m, double scale);
void gsl_my_matrix_add_constant(gsl_matrix * m, double constant);

const int rows=3;
const int columns=3;
int main()
{
   gsl_matrix * A = gsl_matrix_alloc (rows, columns);
   gsl_matrix * B = gsl_matrix_alloc (rows, columns);

   gsl_matrix_set_all(A, 3.0);
   gsl_matrix_memcpy (B, A);
```

```
[A]
[0,0)] = 3      [0,1)] = 3      [0,2)] = 3

 gsl_matrix_mul(A, B);
[0,0)] = 9      [0,1)] = 9      [0,2)] = 9
[1,0)] = 9      [1,1)] = 9      [1,2)] = 9
[2,0)] = 9      [2,1)] = 9      [2,2)] = 9

 gsl_matrix_div(A, B);
[0,0)] = 1      [0,1)] = 1      [0,2)] = 1
[1,0)] = 1      [1,1)] = 1      [1,2)] = 1
[2,0)] = 1      [2,1)] = 1      [2,2)] = 1

 [B]  add constant       1.50
[0,0)] = 4.5    [0,1)] = 4.5    [0,2)] = 4.5
[1,0)] = 4.5    [1,1)] = 4.5    [1,2)] = 4.5
[2,0)] = 4.5    [2,1)] = 4.5    [2,2)] = 4.5

 gsl_matrix_sub(A, B);
[0,0)] = 0      [0,1)] = 0      [0,2)] = 0
[1,0)] = 0      [1,1)] = 0      [1,2)] = 0
[2,0)] = 0      [2,1)] = 0      [2,2)] = 0
```

Πηγαίος κώδικας των σημερινών ασκήσεων υπάρχει στο eclass [lecture 21]

Όψεις Πινάκων με τη GSL (2/2)

Ο πηγαίος κώδικας του παραδείγματος βρίσκεται στο eclass:
GSL-MATRIX-VIEW

```
void gsl_set_matrix_elements(gsl_matrix * m);
void gsl_print_matrix_elements(gsl_matrix * m);
void gsl_print_matrix_view_elements(gsl_matrix_view mv);
int main (void)
{
      gsl_matrix * m = gsl_matrix_alloc (10, 3);
      gsl_set_matrix_elements(m);

      printf("[A] ");
      gsl_print_matrix_elements(m);

      gsl_matrix_view mv = gsl_matrix_submatrix(m, 0, 0, 2, 2);

      gsl_matrix *B = gsl_matrix_alloc (2, 2);
      gsl_matrix_memcpy (B, &mv.matrix);
      gsl_matrix_set (B, 0, 0, 100);

      printf("\n[B] ");
      gsl_print_matrix_elements(B);
      gsl_matrix_free (m);
return 0;
}
```

```
m(0,0) = 0      m(0,1) = 1      m(0,2) = 2
m(1,0) = 1      m(1,1) = 2      m(1,2) = 3
m(2,0) = 2      m(2,1) = 3      m(2,2) = 4
m(3,0) = 3      m(3,1) = 4      m(3,2) = 5
m(4,0) = 4      m(4,1) = 5      m(4,2) = 6
m(5,0) = 5      m(5,1) = 6      m(5,2) = 7
m(6,0) = 6      m(6,1) = 7      m(6,2) = 8
m(7,0) = 7      m(7,1) = 8      m(7,2) = 9
m(8,0) = 8      m(8,1) = 9      m(8,2) = 10
m(9,0) = 9      m(9,1) = 10     m(9,2) = 11

mv(0,0) = 0     mv(0,1) = 1
mv(1,0) = 1     mv(1,1) = 2
```

Ο πηγαίος κώδικας των παραδειγμάτων υπάρχει στο eclass [lecture 20]

Δημιουργία Ταυτοτικού Πίνακα και Πίνακα Μεταθέσεων με τη GSL

Ο πηγαίος κώδικας του παραδείγματος βρίσκεται στο eclass:
GSL-IDENTITYMATRIX-PERMUTATIONS

```
int i;
gsl_matrix * my_identity_matrix = gsl_matrix_alloc (M, M);
gsl_matrix_set_identity (my_identity_matrix);
gsl_permutation * p = gsl_permutation_calloc (M);
gsl_permutation_init (p);
do
{
    printf("\n P E R M U T A T I O N TABLE=");
    gsl_matrix * p_matrix = gsl_matrix_alloc (M, M);
    gsl_permutation_fprintf (stdout, p, " %u");

    for(i=0; i<M; i++){
        int pid=gsl_permutation_get ( p, i);
        gsl_vector * v = gsl_vector_alloc (M);
        gsl_matrix_get_row (v, my_identity_matrix, pid);
        gsl_matrix_set_row(p_matrix,i,v);
        gsl_vector_free (v);
    }
    gsl_print_matrix_elements(p_matrix);
    gsl_matrix_free (p_matrix);
     printf("\n");
}
while (gsl_permutation_next(p) == GSL_SUCCESS);
gsl_matrix_free (my_identity_matrix);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

I_3

$P_{\nu=}\nu!=3!$

```
P E R M U T A T I O N= 0 1 2
      [1]           [0]           [0]
      [0]           [1]           [0]
      [0]           [0]           [1]

P E R M U T A T I O N= 0 2 1
      [1]           [0]           [0]
      [0]           [0]           [1]
      [0]           [1]           [0]

P E R M U T A T I O N= 1 0 2
      [0]           [1]           [0]
      [1]           [0]           [0]
      [0]           [0]           [1]

P E R M U T A T I O N= 1 2 0
      [0]           [1]           [0]
      [0]           [0]           [1]
      [1]           [0]           [0]

P E R M U T A T I O N= 2 0 1
      [0]           [0]           [1]
      [1]           [0]           [0]
      [0]           [1]           [0]

P E R M U T A T I O N= 2 1 0
      [0]           [0]           [1]
      [0]           [1]           [0]
      [1]           [0]           [0]
```

Πηγαίος κώδικας των σημερινών ασκήσεων υπάρχει στο eclass [lecture 20]

Ιδιοτιμές, Ιδιοδιανύσματα Πίνακα

Ο πηγαίος κώδικας του παραδείγματος βρίσκεται στο eclass:
GSL-EIGENVALUES-EIGENVECTORS

```
#include <gsl/gsl_eigen.h>

int main()
{
   int N=3;
   int i;
   double a_data[]={2,6,4,6,3,2,4,2,-2};
   gsl_matrix_view m=gsl_matrix_view_array(a_data,N,N);
   gsl_vector *eval = gsl_vector_alloc (N);
   gsl_matrix *evec = gsl_matrix_alloc (N, N);

   gsl_eigen_symmv_workspace * w = gsl_eigen_symmv_alloc (N);
   gsl_eigen_symmv(&m.matrix, eval, evec, w);
   gsl_eigen_symm_free (w);

   for (i = 0; i < N; i++)
   {
      double eval_i= gsl_vector_get (eval, i);
      gsl_vector_view evec_i= gsl_matrix_column (evec, i);
      printf ("eigenvalue = %g\n", eval_i);
      printf ("eigenvector = \n");
      gsl_vector_fprintf (stdout, &evec_i.vector, "%g");
    }
   return 0;
}
```

```
eigenvalue = 10
eigenvector =
0.666667
0.666667
0.333333
eigenvalue = -5
eigenvector =
0.666667
-0.333333
-0.666667
eigenvalue = -2
eigenvector =
0.333333
-0.666667
0.666667
```

Ο πηγαίος κώδικας των παραδειγμάτων υπάρχει στο eclass [lecture 20]

2η Δραστηριότητα με τη GSL: Επίλυση Γραμμικού Συστήματος με τη Μέθοδο απαλοιφής Gauss

Ένα γραμμικό σύστημα στην γενική του μορφή n-εξισώσεων με n-αγνώστους:

$$\begin{aligned}
\alpha_{11}x_1 + \alpha_{12}x_2 + \ldots + \alpha_{1n}x_n &= b_1 \\
\alpha_{21}x_1 + \alpha_{22}x_2 + \ldots + \alpha_{2n}x_n &= b_2 \\
\vdots \qquad\qquad & \ldots \qquad\qquad \vdots \\
\alpha_{n1}x_1 + \alpha_{n2}x_2 + \ldots + \alpha_{nn}x_n &= b_n
\end{aligned}$$

$$\underbrace{\begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots\alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \ldots\alpha_{2n} \\ \vdots & & \ldots\ \vdots \\ \alpha_{n1} & \alpha_{m2} & \ldots\alpha_{nn} \end{bmatrix}}_{A} \quad \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}}_{b}$$

Επαυξημένος πίνακας $[A|\mathbf{b}]_{nxn+1}$

$$\left[\begin{array}{ccc|c} \alpha_{11} & \alpha_{12} & \ldots\alpha_{1n} & b_1 \\ \alpha_{21} & \alpha_{22} & \ldots\alpha_{2n} & b_2 \\ \vdots & & \ldots\ \vdots & \ldots \\ \alpha_{n1} & \alpha_{m2} & \ldots\alpha_{nn} & b_n \end{array}\right]$$

Echelon form matrix

$$\left[\begin{array}{ccc|c} \alpha_{11} & \alpha_{12} & \ldots\alpha_{1n} & b_{1'} \\ 0 & \alpha_{22} & \ldots\alpha_{2n} & b_{2'} \\ \vdots & & \ldots\ \vdots & \ldots \\ 0 & 0 & \ldots\alpha_{nn} & b_{n'} \end{array}\right]$$

Προς τα πίσω αντικατάσταση

$$x_n = b_{n'} / \alpha_{nn'} \quad \vdots$$

2η Δραστηριότητα:
Πηγαίος κώδικας πρώτης έκδοσης

Παραδοτέο πρώτης έκδοσης:

```
#include <stdlib.h>
#include <time.h>
#include<unistd.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_permutation.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_rng.h>
void initialize_linear_algebra_system (gsl_matrix * a, gsl_vector * b);
void create_augmented_matrix (gsl_matrix * a, gsl_vector * b,gsl_matrix * aug
void print_augmented_matrix(gsl_matrix * aug_matrix,gsl_matrix * a, gsl_vect
gsl_rng * seed_gsl_mt_rng();
int main()
{
   gsl_matrix * a;
   gsl_vector * x;
   gsl_vector * b;
   gsl_matrix * aug_matrix;

   int n;
   printf("Number of equations: ");
   scanf("%d",&n);
   a = gsl_matrix_alloc (n, n);
   x = gsl_vector_alloc (n);
   b = gsl_vector_alloc (n);
   aug_matrix=gsl_matrix_alloc (n, n+1);

   initialize_linear_algebra_system (a, b);
   create_augmented_matrix (a, b, aug_matrix );
   print_augmented_matrix(aug_matrix,a,b);


    gsl_matrix_free (a);
    gsl_vector_free (x);
    gsl_vector_free (b);
    gsl_matrix_free (aug_matrix);
   return 0;
```

```
void create_augmented_matrix (gsl_matrix * a, gsl_vector *
   int i;
   for (i = 0; i < a->size1; i++){
      gsl_vector * tmp = gsl_vector_alloc (a->size1);
      gsl_matrix_get_col (tmp, a, i);
      gsl_matrix_set_col(aug_matrix,i,tmp);
      gsl_vector_free (tmp);

   }
    gsl_matrix_set_col(aug_matrix,aug_matrix->size2-1,b);
}
```

2η Δραστηριότητα:
Παραδοτέο πρώτης έκδοσης

```
Number of equations: 7

 [A]
     2.0000          8.0000          2.0000          1.0000          7.0000          5.0000          9.0000
     2.0000          4.0000          1.0000          9.0000          2.0000          3.0000          1.0000
     6.0000          8.0000          4.0000          3.0000          5.0000          1.0000          4.0000
     2.0000          8.0000          9.0000          1.0000          4.0000          7.0000          3.0000
     7.0000          7.0000          1.0000          8.0000          8.0000          2.0000          7.0000
     2.0000          5.0000          7.0000          8.0000          9.0000          3.0000          7.0000
     2.0000          5.0000          3.0000          4.0000          7.0000          8.0000          6.0000


 [b]
    2.0000
    2.0000
    9.0000
    6.0000
    6.0000
    4.0000
    6.0000


 [A|b]
     2.0000          8.0000          2.0000          1.0000          7.0000          5.0000          9.0000          2.0000
     2.0000          4.0000          1.0000          9.0000          2.0000          3.0000          1.0000          2.0000
     6.0000          8.0000          4.0000          3.0000          5.0000          1.0000          4.0000          9.0000
     2.0000          8.0000          9.0000          1.0000          4.0000          7.0000          3.0000          6.0000
     7.0000          7.0000          1.0000          8.0000          8.0000          2.0000          7.0000          6.0000
     2.0000          5.0000          7.0000          8.0000          9.0000          3.0000          7.0000          4.0000
     2.0000          5.0000          3.0000          4.0000          7.0000          8.0000          6.0000          6.0000
```

2η Δραστηριότητα: Πηγαίος κώδικας δεύτερης έκδοσης

```c
#include <gsl/gsl_vector.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_blas.h>
void initialize_linear_algebra_system_mt_rng (gsl_matrix * a, gsl_vector * b);
void initialize_linear_algebra_system_from_array(gsl_matrix * a, gsl_vector * b);
void create_augmented_matrix (gsl_matrix * a, gsl_vector * b,gsl_matrix * aug_matrix );
void  gauss_elimination(gsl_matrix * aug_matrix);
void gauss_substitution (gsl_matrix * echelon, gsl_vector * x);
void verify_solution(gsl_matrix * a, gsl_vector * x);

gsl_rng * seed_gsl_mt_rng();
void  print_lasystem_elements(gsl_matrix * a, gsl_vector * b,gsl_matrix * aug_matrix,
                        gsl_matrix * echelon_matrix,gsl_vector * x);
void print_gsl_array(gsl_matrix * m);
void print_gsl_vector(gsl_vector * v);
int main()
{
   gsl_matrix * a,* aug_matrix,* echelonform_matrix;
   gsl_vector * x,*b;
   int n;
   printf("Number of equations: ");
   scanf("%d",&n);
   a = gsl_matrix_alloc (n, n);
   x = gsl_vector_alloc (n);
   b = gsl_vector_alloc (n);
   aug_matrix=gsl_matrix_alloc (n, n+1);
   echelonform_matrix=gsl_matrix_alloc (n, n+1);

   initialize_linear_algebra_system_mt_rng (a, b);
   create_augmented_matrix (a, b, aug_matrix );
   gsl_matrix_memcpy (echelonform_matrix, aug_matrix);
   gauss_elimination(echelonform_matrix);
   gauss_substitution (echelonform_matrix,  x);
   print_lasystem_elements(a,b,aug_matrix,echelonform_matrix,x);
   verify_solution(a, x);

    gsl_matrix_free (a); gsl_vector_free (x); gsl_vector_free (b);
    gsl_matrix_free (aug_matrix); gsl_matrix_free(echelonform_matrix);
   return 0;
```

```c
void initialize_linear_algebra_system_mt_rng (gsl_matrix * a

    int i,j;
     gsl_rng *mt_rng=seed_gsl_mt_rng();

     int mt_res;
      for (i = 0; i < a->size1; i++){
        for (j = 0; j < a->size2; j++) {
                  mt_res = gsl_rng_uniform (mt_rng) * 999+1;
                  gsl_matrix_set (a, i, j,mt_res);
        }
      }

       for (i = 0; i < b->size; i++) {
           mt_res = gsl_rng_uniform (mt_rng) * 999+1;
           gsl_vector_set (b, i, mt_res);
       }
}
```

```c
void initialize_linear_algebra_system_from_array(gsl_matrix * a, gsl_vector * b){

  double       a_data[]={2,1,3,4,4,16,1,1,2};
  double       b_data[]={12,16,8};

  gsl_matrix_view mv=gsl_matrix_view_array(a_data,a->size1,a->size2);
  gsl_matrix_memcpy (a, &mv.matrix);


  gsl_vector_view vv = gsl_vector_view_array (b_data, b->size);
  gsl_vector_memcpy (b, &vv.vector);
```

2η Δραστηριότητα: Παραδοτέο δεύτερης έκδοσης

Παραδοτέο δεύτερης έκδοσης:

```
Number of equations: 7

 [A]
     5.0000         1.0000         7.0000         2.0000         1.0000         9.0000         2.0000
     8.0000         4.0000         9.0000         1.0000         9.0000         4.0000         1.0000
     5.0000         2.0000         8.0000         7.0000         1.0000         5.0000         2.0000
     8.0000         7.0000         9.0000         6.0000         1.0000         3.0000         9.0000
     9.0000         8.0000         8.0000         4.0000         2.0000         2.0000         8.0000
     1.0000         3.0000         7.0000         9.0000         2.0000         7.0000         5.0000
     6.0000         9.0000         3.0000         2.0000         4.0000         1.0000         6.0000

 [b]
    7.0000
    2.0000
    6.0000
    9.0000
    1.0000
    4.0000
    2.0000


 [x]
   23.6908
  -25.8555
  -26.6475
   13.0867
   12.5879
    3.4826
   15.4149


V E R I F I C A T I O N: [A] * [x] = [b]
    7.0000
    2.0000
    6.0000
    9.0000
    1.0000
    4.0000
    2.0000
```

2η Δραστηριότητα: Πηγαίος κώδικας τρίτης έκδοσης

```c
#include <gsl/gsl_permutation.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_blas.h>
void initialize_linear_algebra_system_mt_rng (gsl_
void initialize_linear_algebra_system_from_array(
void create_augmented_matrix (gsl_matrix * a, gs
void  gauss_elimination(gsl_matrix * aug_matrix,i
void gauss_substitution (gsl_matrix * echelon, gsl_
void verify_solution(gsl_matrix * a, gsl_vector * x);

gsl_rng * seed_gsl_mt_rng();
void  print_lasystem_elements(gsl_matrix * a, gsl_
                        gsl_matrix * echelon_matrix,gsl
void print_gsl_array(gsl_matrix * m);
void print_gsl_vector(gsl_vector * v);
void bench_gauss_elimination(int n,int *flop);

int main()
{
   int i,flop;
   for(i=2;i<=300;i++){
         flop=0;
         printf("Linear system (%d,%d)",i,i);
         bench_gauss_elimination(i,&flop);
         printf("\t required number of flop:%d \n",flop
         printf("%d \n",flop);
   }
   return 0;
}
```

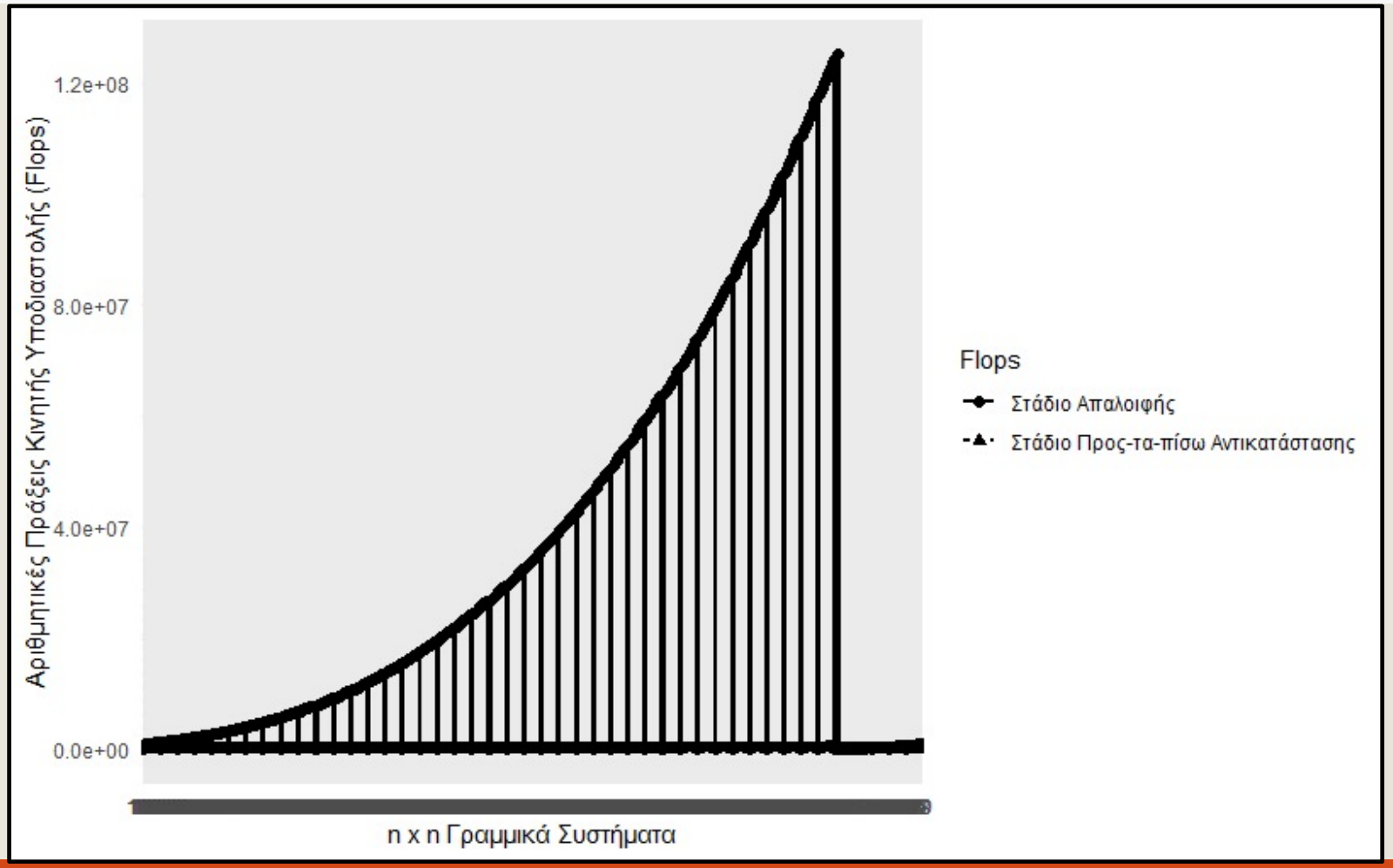
```c
void bench_gauss_elimination(int n,int *flop){

   gsl_matrix * a,* aug_matrix,* echelonform_matrix;
   gsl_vector * x,*b;

   a = gsl_matrix_alloc (n, n);
   x = gsl_vector_alloc (n);
   b = gsl_vector_alloc (n);
   aug_matrix=gsl_matrix_alloc (n, n+1);
   echelonform_matrix=gsl_matrix_alloc (n, n+1);

   initialize_linear_algebra_system_mt_rng (a, b);
   create_augmented_matrix (a, b, aug_matrix );
   gsl_matrix_memcpy (echelonform_matrix, aug_matrix);
   gauss_elimination(echelonform_matrix,flop);
   gauss_substitution (echelonform_matrix,  x,flop);
    gsl_matrix_free (a); gsl_vector_free (x); gsl_vector_free (b);
    gsl_matrix_free (aug_matrix); gsl_matrix_free(echelonform_matrix);
}
```

2η Δραστηριότητα - Τρίτη Έκδοση: Υπολογισμός Επεξεργαστικής Ισχύς Point Operations (FLOP)



Επίλυση Γραμμικού Συστήματος με τη GSL

Η επίλυση ενός γραμμικού συστήματος γίνεται μέσω της διεπαφής:
#include <gsl/gsl_linalg.h>

- Δημιουργία Πίνακα Μετάθεσης:

```
gsl_permutation * p = gsl_permutation_alloc (b->size);
```

- Παραγοντοποίηση LU του Πίνακα και υπολογισμός ορίζουσας:

```
int swaps;
gsl_linalg_LU_decomp(A,p,&swaps);
gsl_linalg_LU_det(gsl matrix * LU, swaps);
```

- Επίλυση του Γραμμικού Συστήματος:

```
    gsl_linalg_LU_solve(A,p,b,x);
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

Αρχικός κατάλογος » ΔΙΑΛΕΞΕΙΣ 2021 » lecture20

Τύπος	Όνομα
	GSL - Δραστηριότητα 2
	GSL - Πηγαίος Κώδικας Παραδειγμάτων Θεωρίας
	Διαφάνειες

Προς τα πίσω αντικατάσταση

FLOP ως μονάδα μέτρησης της απαιτούμενης επεξεργαστικής ισχύς σε ένα υπολογιστικό πρόβλημα. Παράδειγμα:

Echelon form matrix

$$\left[\begin{array}{ccc|c} \alpha_{11} & \alpha_{12} \ldots & \alpha_{1n} & b_{1'} \\ 0 & \alpha_{22} \ldots & \alpha_{2n} & b_{2'} \\ \vdots & \ldots & \vdots & \ldots \\ 0 & 0 \ldots & \alpha_{nn} & b_{n'} \end{array}\right]$$

FLOPS - Προς τα πίσω αντικατάσταση της μεθόδου απαλοιφής του Gauss

$$X_n = b_{n'} / \alpha_{nn'}$$

$$X_{n-1} = (b_{n-1'} - X_n * \alpha_{n-1n}) / \alpha_{n-1',n-1}$$

$$\vdots$$

FLOPS - Προς τα πίσω αντικατάσταση της μεθόδου απαλοιφής του Gauss

```
ls (2,2)        flop for bsub :4
ls (3,3)        flop for bsub :9
ls (4,4)        flop for bsub :16
ls (5,5)        flop for bsub :25
ls (6,6)        flop for bsub :36
ls (7,7)        flop for bsub :49
ls (8,8)        flop for bsub :64
ls (9,9)        flop for bsub :81
ls (10,10)      flop for bsub :100
```

$$\text{FLOP} = 1 + \sum_{i=2}^{i=n} [(i-1) * 2 + 1]$$