ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ
ECE_Y215: ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

2η Δραστηριότητα με χρήση της Βιβλιοθήκης GSL (GNU Scientific Library): Επίλυση Γραμμικών Συστημάτων με χρήση της Μεθόδου Απαλοιφής του Gauss

Εισαγωγή

Σκοπός της παρούσας δραστηριότητας είναι η εξοικείωση με την βιβλιοθήκη GSL [1] και ειδικότερα με τις διεπαφές διαχείρισης πινάκων (*#include <gsl/gsl_matrix.h>*), διανυσμάτων (*#include <gsl/gsl_vector.h>*) και βασικών συναρτήσεων γραμμικής άλγεβρας (*#include <gsl/gsl_blas.h>*).

Το πρόβλημα της επίλυσης γραμμικών συστημάτων είναι συχνό στον επιστημονικό υπολογισμό και σχετίζεται εκτός άλλων με υπολογιστικά προβλήματα τεχνητής νοημοσύνης, μοντελοποίησης φυσικών φαινομένων, επιστήμες διαχείρισης μεγάλου όγκου δεδομένων. Ένα γραμμικό σύστημα στην γενική του μορφή *n*-εξισώσεων με *n*-αγνώστους φαίνεται παρακάτω:

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

Σχήμα 1: Γραμμικό σύστημα στην γενική του μορφή *n*-εξισώσεων με *n*-αγνώστους.

Το παραπάνω σύστημα μπορεί να γραφεί σε μορφή πινάκων και διανυσμάτων ως εξής (**Σχήμα 2**): πίνακας συντελεστών των αγνώστων [A] όπου a_{ij} (με i,j =1,2,3,…,n), διάνυσμα των αγνώστων [x] όπου x_i (i=1,2,3,…,n) και διάνυσμα σταθερών όρων [b] όπου b_i (i=1,2,3,…,n).

$$\underbrace{\begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots\alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \ldots\alpha_{2n} \\ \vdots & & \ldots \ \vdots \\ \alpha_{n1} & \alpha_{n2} & \ldots\alpha_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}}_{b}$$

Σχήμα 2: Γραμμικό σύστημα σε μορφή πινάκων και διανυσμάτων.

Ζητούμενα

Ένας από τους τρόπους επίλυσης των γραμμικών συστημάτων είναι η μέθοδος απαλοιφής του Gauss [2]. Δημιουργήστε ένα πρόγραμμα στη C με χρήση της βιβλιοθήκης GSL το οποίο θα υλοποιεί την μέθοδο απαλοιφής Gauss.

Παραδοτέα

Η δραστηριότητα περιλαμβάνει τρεις εκδόσεις. Στο Παράρτημα δίνεται ένας βασικός σκελετός κώδικα για κάθε έκδοση, ο οποίος περιλαμβάνει τις δηλώσεις των συναρτήσεων τις οποίες πρέπει να υλοποιήσετε. Ωστόσο, είστε ελεύθεροι να δημιουργήσετε ανεξάρτητα από τον ενδεικτικό κώδικα όποιο κώδικα εσείς επιθυμείτε.

Έκδοση 1: Αρχικοποίηση του πίνακα [*A*] (συντελεστές των αγνώστων) και του διανύσματος [*b*] (σταθεροί όροι). Δημιουργία του επαυξημένου πίνακα [*A*|*b*].

Ο σκοπός της πρώτης έκδοσης είναι να αποκτήσετε εξοικείωση με την μετατροπή ενός συστήματος γραμμικών εξισώσεων σε μορφή πινάκων και διανυσμάτων με τη GSL. Συγκεκριμένα για ένα γραμμικό σύστημα με *n* εξισώσεις και *n* αγνώστους (τον αριθμό *n* θα τον εισάγει ο χρήστης), αρχικοποιήστε τον πίνακα στοιχείων [*A*] και σταθερών όρων [*b*] μέσω μίας γεννήτριας τυχαίων ακεραίων αριθμών στο διάστημα [1,1000] (αποφεύγουμε την αρχικοποίηση με μηδενικά στοιχεία, ώστε να αποφύγουμε την πιθανότητα το σύστημα να είναι αόριστο ή να έχει άπειρες λύσεις).

Στη συνέχεια δημιουργήστε και εκτυπώστε τον επαυξημένο πίνακα [*A*|***b***] ο οποίος θα προκύπτει από τον πίνακα *A* αν του προσθέσουμε μια επιπλέον στήλη που είναι το διάνυσμα σταθερών όρων [*b*]. Στην **Εικόνα 1** του Παραρτήματος, υπάρχει ενδεικτικός πηγαίος κώδικας της πρώτης έκδοσης. Στον πηγαίο κώδικα φαίνονται οι δηλώσεις των συναρτήσεων χωρίς την υλοποίησή τους[1]. Βασικές συναρτήσεις της πρώτης έκδοσης είναι:

- void **initialize_linear_algebra_system** (*gsl_matrix * a, gsl_vector * b*);
- void **create_augmented_matrix** (*gsl_matrix * a, gsl_vector * b, gsl_matrix * augm_ma*);
- void **print_augmented_matrix** (*gsl_matrix * augm_matrix*);

Έκδοση 2: Επίλυση του γραμμικού συστήματος με τη μέθοδο Gauss

Ο σκοπός της δεύτερης έκδοσης είναι η επίλυση ενός γραμμικού συστήματος με τη μέθοδο Gauss. Η δεύτερη έκδοση θα επεκτείνει την πρώτη έκδοση προσθέτοντας δύο επιπλέον στάδια: *α)* το στάδιο της «απαλοιφής» (elimination) των αγνώστων (όπου ο πίνακας συντελεστών [*A*] τριγωνοποιείται και ταυτόχρονα το διάνυσμα σταθερών [*b*] μετατρέπεται καταλλήλως, και *β)* το στάδιο της προς-τα-πίσω αντικατάστασης (back substitution).

Κατά το στάδιο της προς-τα-πίσω αντικατάστασης του αλγορίθμου, βρίσκουμε την τελευταία εξίσωση την τιμή του x_n, η οποία στη συνέχεια αντικαθίσταται για όλες τις επόμενες εξισώσεις μέχρι να βρεθεί η τιμή x_1 και έτσι να λυθεί το γραμμικό σύστημα.

Επίσης, η δεύτερη έκδοση του προγράμματος επαληθεύει την ορθότητα της λύσης κάνοντας χρήση της συνάρτησης *gsl_blas_dgemv* της *gsl_blas.h*. Στην **Εικόνα 2** του Παραρτήματος υπάρχει ενδεικτικός πηγαίος κώδικας της δεύτερης έκδοσης. Στον πηγαίο κώδικα φαίνονται οι δηλώσεις των συναρτήσεων χωρίς την υλοποίησή τους[2]. Βασικές συναρτήσεις της δεύτερης έκδοσης είναι:

- void **gauss_elimination** (*gsl_matrix * augmented_matrix*);
- void **gauss_substitution** (*gsl_matrix * echelon, gsl_vector * x*);
- void **verify_solution** (*gsl_matrix * a, gsl_vector * x*);

---

[1] *Σημείωση: Η ονομασία των συναρτήσεων καθώς και τα ορίσματα είναι ενδεικτικά. Μπορείτε να επιλέξετε οποιοδήποτε ονομασία συναρτήσεων επιθυμείτε.*

Έκδοση 3: Καταμέτρηση των αριθμητικών πράξεων κινητής υποδιαστολής (*flop*) για διαφορετικά σενάρια επίλυσης γραμμικών συστημάτων με την μέθοδο απαλοιφής του Gauss

Ο σκοπός της τρίτης έκδοσης είναι να αποκτήσουμε εμπειρική γνώση πάνω σε θέματα απαιτούμενης επεξεργαστικής ισχύς σχετικά με αριθμητικές πράξεις κινητής υποδιαστολής (*floating point operations-flop*[3]). Η αναφορά στα flop ως μονάδα μέτρησης της απαιτούμενης επεξεργαστικής ισχύς, έχει ιδιαίτερη σημασία στον τομέα των επιστημονικών υπολογισμών, όπου γίνεται εκτεταμένη χρήση πράξεων κινητής υποδιαστολής.

Για το λόγο αυτό, δημιουργήστε μια μέθοδο *bench_gauss_elimination* η οποία θα καταγράφει το σύνολο των αριθμητικών πράξεων κινητής υποδιαστολής (*flop*) που χρειάζονται σύμφωνα με την μέθοδο απαλοιφής του Gauss για την επίλυση καθενός από τα γραμμικά συστήματα 2x2, 3x3,… 500x500. Καλέστε επαναληπτικά την αρχικοποίηση και επίλυση γραμμικού συστήματος που υλοποιήσατε στην δεύτερη έκδοση της δραστηριότητας.

Καταγράψτε ξεχωριστά τον αριθμό των flop και για την τριγωνοποίηση του επαυξημένου πίνακα και για την προς τα πίσω αντικατάσταση. Σε ποιο συμπέρασμα μας οδηγούν οι μετρήσεις μας σχετικά με την απαιτούμενη επεξεργαστική ισχύ της μεθόδου απαλοιφής του Gauss σε σχέση με την αύξηση του αριθμού των γραμμικών εξισώσεων;

Στην Εικόνα 3 του Παραρτήματος υπάρχει ενδεικτικός πηγαίος κώδικας της τρίτης έκδοσης. Στον πηγαίο κώδικα φαίνονται οι δηλώσεις των συναρτήσεων χωρίς την υλοποίησή τους[3]. Βασικές συναρτήσεις της τρίτης έκδοσης είναι:

- void **bench_gauss_elimination**(*int n,int *flop_elim,int *flop_bsub*);
- void **gauss_elimination** (*gsl_matrix * augmented_matrix, int *flop_elim*);
- void **gauss_substitution** (*gsl_matrix * echelon, gsl_vector * x, , int *flop_bsub*);

Καταγράψτε τις παρατηρήσεις σας σε μια μικρή έκθεση σχετικά με το υπολογιστικό κόστος (*αριθμός flop*) της μεθόδου απαλοιφής του Gauss που εξετάσαμε για την επίλυση ενός γραμμικού συστήματος. Αναλογιστείτε ότι τυπικά επιστημονικά μοντέλα διαχειρίζονται δεκάδες χιλιάδες εξισώσεις και μεταβλητές. Ανεβάστε τις εκδόσεις του κώδικα σας μαζί με την αναφορά σας στο e-class σε μορφή συμπιεσμένου αρχείου και τίτλο τον αριθμό μητρώου σας.

Αναφορές

[1] https://www.gnu.org/software/gsl
[2] https://en.wikipedia.org/wiki/Gaussian_elimination
[3] https://en.wikipedia.org/wiki/FLOPS

Σημείωση: Η ονομασία των συναρτήσεων καθώς και τα ορίσματα είναι ενδεικτικά. Μπορείτε να επιλέξετε οποιοδήποτε ονομασία συναρτήσεων επιθυμείτε.

Παράρτημα
1η Έκδοση

```c
#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_matrix );
void print_augmented_matrix(gsl_matrix * aug_matrix,gsl_matrix * a, gsl_vector * b);
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;
```

```
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
```

Εικόνα 1. Ενδεικτικός πηγαίος κώδικας και παράδειγμα εκτέλεσης της πρώτης έκδοσης.

2η Έκδοση

```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;
```

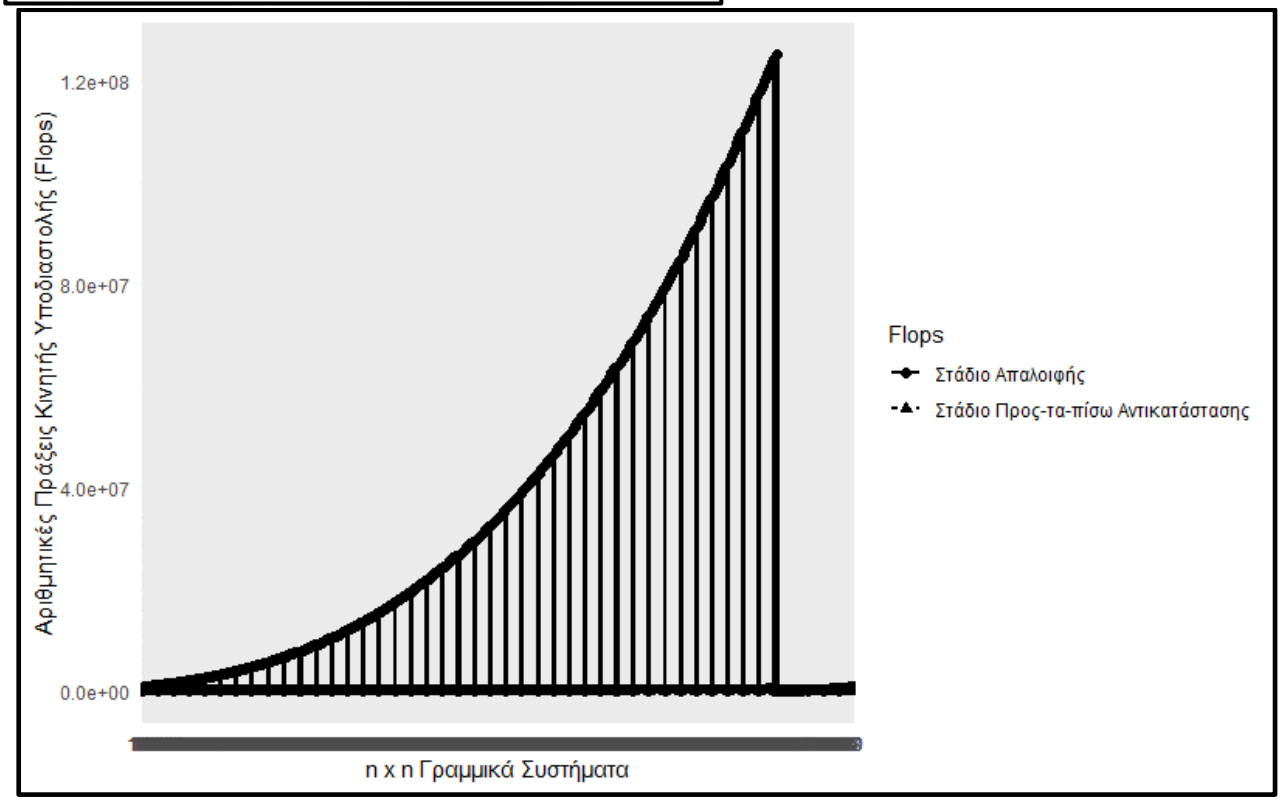
```
Number of equations: 7

 [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
```

Εικόνα 2. Ενδεικτικός πηγαίος κώδικας και παράδειγμα εκτέλεσης δεύτερης έκδοσης.

3η Έκδοση

```c
void bench_gauss_elimination(int n,int *flop_elim,int *flop_bsub);

int main()
{
   int i,flop_elim,flop_bsub;
    printf("Linear System \t  Elimination(flop) \t  Back-Substituion(flop) \n" );
   for(i=2;i<=500;i++){
         flop_elim=0;
         flop_bsub=0;
         bench_gauss_elimination(i,&flop_elim, &flop_bsub);
         printf(" [%2d,%2d] \t\t %d \t\t\t %d \n",i,i,flop_elim,flop_bsub);

   }
   return 0;
}

void bench_gauss_elimination(int n,int *flop_elim,int *flop_bsub){

   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);
```

Linear System	Elimination(flop)	Back-Substituion(flop)
[2, 2]	7	4
[3, 3]	25	9
[4, 4]	58	16
[5, 5]	110	25
[6, 6]	185	36
[7, 7]	287	49
[8, 8]	420	64
[9, 9]	588	81
[10,10]	795	100
[11,11]	1045	121
[12,12]	1342	144
[13,13]	1690	169
[14,14]	2093	196
[15,15]	2555	225
[16,16]	3080	256
[17,17]	3672	289
[18,18]	4335	324
[19,19]	5073	361
[20,20]	5890	400
[21,21]	6790	441
[22,22]	7777	484
[23,23]	8855	529
[24,24]	10028	576
[25,25]	11300	625
[26,26]	12675	676
[27,27]	14157	729
[28,28]	15750	784
[29,29]	17458	841
[30,30]	19285	900
[31,31]	21235	961
[32,32]	23312	1024
[33,33]	25520	1089



Εικόνα 3. Ενδεικτικός πηγαίος κώδικας και παράδειγμα εκτέλεσης τρίτης έκδοσης. Η γραφική παράσταση δημιουργήθηκε βάση των αποτελεσμάτων εκτέλεσης του ενδεικτικού κώδικα για *n*=500.