sdi2000014 – Θεοδώρα Αρχοντάκη

Εξήγηση κώδικα για την μέθοδο ESOR:

Έχουμε τα παρακάτω αρχεία:

- ask2_ESOR.cpp: περιέχει τις συναρτήσεις για την επίλυση ενός γραμμικού συστήματος με την μέθοδο ESOR. Η συνάρτηση «ESOR» εκτελεί τον αλγόριθμο επίλυσης του γραμμικού συστήματος. Έχουμε ένα loop που τρέχει όσο το error > epsilon και σε κάθε επανάληψη υπολογίζει το νέο υπόλοιπο, z, p και έτσι ανανεώνει την λύση x. Στο τέλος της κάθε επανάληψης υπολογίζει το νέο error και αυξάνει τον αριθμό των επαναλήψεων.
- Process_data.cpp: έχει ακριβώς την ίδια υλοποίηση με το αντίστοιχο αρχείο της προηγούμενης εργασίας. Στο συγκεκριμένο αρχείο έχουμε επιπλέον την συνάρτηση "create5_diagonal" που δημιουργεί έναν τυχαίο πενταδιαγώνιο πίνακα και την συνάρτηση "createA" η οποία δημιουργεί έναν πίνακα A με συγκεκριμένα α,β,γ,δ. Τα α,β,γ,δ έχουν οριστεί με define στην αρχή του αρχείου. Ο χρήστης μπορεί να δώσει τα δεδομένα με τους παρακάτω τρόπους:
    - Μπορεί να δώσει ένα προς ένα τα στοιχεία του A και του b από την γραμμή εντολών.
    - Μπορεί το πρόγραμμα να διαβάσει τους πίνακες από ένα αρχείο κειμένου.
    - Μπορεί το πρόγραμμα να ζητήσει από τον χρήστη απλά το μέγεθος του πίνακα και να φτιάξει έναν τυχαίο πίνακα ή να χρησιμοποιήσει έναν ήδη έτοιμο (φτιάχνει με την συνάρτηση "createA" τον A με α=0.1,β=0.2,γ=0.3,δ=0.4).
- main.cpp: περιέχει την main του προγράμματος η οποία καλεί την συνάρτηση για την επίλυση ενός γραμμικού συστήματος μέσα σε μία διπλή λούπα που διατρέχει όλες τις τιμές των ω, τ ώστε να βρει τις βέλτιστες τιμές.
- Process_data.h: περιέχει τους ορισμούς των συναρτήσεων που βρίσκονται στο αρχείο Process_data.cpp.
- Functions.h: περιέχει τους ορισμούς των συναρτήσεων που βρίσκονται στο αρχείο ask2_ESOR.cpp.

Το αρχείο κειμένου από το οποίο το πρόγραμμα μπορεί να διαβάζει τα δεδομένα πρέπει να έχει την παρακάτω μορφή. Στην πρώτη γραμμή θα έχει την τιμή του n. Για τις επόμενες n γραμμές θα είναι ο πίνακας A στη συνέχεια θα υπάρχει μία κενή γραμμή και στο τέλος ο πίνακας b. π.χ. αν n = 3 το αρχείο κειμένου θα πρέπει να είναι:

3

$a_{11}\ a_{1_2}\ a_{1_3}$

$a_{2_1}\ a_{2_2}\ a_{2_3}$

$a_{3_1}\ a_{3_2}\ a_{3_3}$

$b_1\ b_2\ b_3$

Εξήγηση κώδικα για την μέθοδο PSD:

Έχουμε τα παρακάτω αρχεία:

- ask2_PSD.cpp: περιέχει τις συναρτήσεις για την επίλυση ενός γραμμικού συστήματος με την μέθοδο PSD. Η συνάρτηση «psdNiethammer» εκτελεί τον αλγόριθμο επίλυσης του γραμμικού συστήματος. Έχουμε ένα loop που τρέχει όσο το error > epsilon και σε κάθε επανάληψη υπολογίζει το νέο υπόλοιπο, z, alpha, beta και έτσι ανανεώνει την λύση x. Στο τέλος της κάθε επανάληψης υπολογίζει το νέο error και αυξάνει τον αριθμό των επαναλήψεων.
- Process_data.cpp: έχει ακριβώς την ίδια υλοποίηση με το αντίστοιχο αρχείο της ESOR μεθόδου. Τα α,β,γ,δ ορίζονται με define στην αρχή του αρχείου. Και έχουν τιμές: α=0.1, β=0.2, γ=0.3, δ=0.4.
- main.cpp: περιέχει την main του προγράμματος η οποία καλεί την συνάρτηση για την επίλυση ενός γραμμικού συστήματος μέσα σε μία διπλή λούπα που διατρέχει όλες τις τιμές των ω, τ ώστε να βρει τις βέλτιστες τιμές.
- Process_data.h: περιέχει τους ορισμούς των συναρτήσεων που βρίσκονται στο αρχείο Process_data.cpp.
- Functions.h: περιέχει τους ορισμούς των συναρτήσεων που βρίσκονται στο αρχείο ask2_PSD.cpp.

Οδηγίες μεταγλώττισης προγραμμάτων:

- Για την ESOR μέθοδο: πάμε στον φάκελο «ESOR» κάνουμε make και δημιουργείται το εκτελέσιμο αρχείο με όνομα «esor». Εκτελούμε το esor "./esor" το πρόγραμμα ρωτάει τον χρήστη αρχικά αν θέλει να δώσει το διάνυσμα b, αν δεν το δώσει υπολογίζεται το b με δεδομένη λύση $x^T = (1,1,\dots 1)$. Μετά ρωτάει με ποιον τρόπο ο χρήστης θέλει να δώσει τα δεδομένα. Υπάρχουν οι εξής τρόποι:
  i. Να δώσει ο χρήστης τα στοιχεία των πινάκων ένα προς ένα
  ii. Να δημιουργηθούν τυχαία δεδομένα ή να χρησιμοποιηθεί ένας ήδη υπάρχον πίνακας (με α=0.1,β=0.2,γ=0.3,δ=0.4).
  iii. Να διαβαστούν τα δεδομένα από ένα αρχείο κειμένου.

  Στη συνέχεια λύνει το γραμμικό σύστημα Ax = b για όλες τις τιμές του τ και του ω, βρίσκει την βέλτιστη τιμή του τ και του ω και υπολογίζει το CPU time.

- Για την PSD μέθοδο: πάμε στον φάκελο «PSD» κάνουμε make και δημιουργείται το εκτελέσιμο αρχείο με όνομα «psd». Εκτελούμε το psd "./psd" το πρόγραμμα ρωτάει τον χρήστη αρχικά αν θέλει να δώσει το διάνυσμα b, αν δεν το δώσει υπολογίζεται το b με δεδομένη λύση $x^T = (1,1,\dots 1)$. Μετά ρωτάει με ποιον τρόπο ο χρήστης θέλει να δώσει τα δεδομένα. Υπάρχουν οι εξής τρόποι:
  i. Να δώσει ο χρήστης τα στοιχεία των πινάκων ένα προς ένα
  ii. Να δημιουργηθούν τυχαία δεδομένα ή να χρησιμοποιηθεί ένας ήδη υπάρχον πίνακας (με α=0.1,β=0.2,γ=0.3,δ=0.4).
  iii. Να διαβαστούν τα δεδομένα από ένα αρχείο κειμένου.

  Στη συνέχεια λύνει το γραμμικό σύστημα Ax = b για όλες τις τιμές του τ και του ω, βρίσκει την βέλτιστη τιμή του τ και του ω και υπολογίζει το CPU time.

Για τον αλγόριθμο ESOR:

Πίνακας (**Εφαρμογή** 1)

Επίλυση του $\mathrm{Ax} = \mathrm{b}$ ESOR									
Διάσταση A	**Παράμετροι**				**Βέλτιστη τιμή**	**Βέλτιστη τιμή**	**Φασματική ακτίνα**	**Αριθμός επαναλήψεων**	**Χρόνος εκτέλεσης**
	α	β	γ	δ	τ_b	ω_b	$\rho(G(\tau_b,\omega_b))$	*itcount*	*cputime*
$n = 10^2$	0.1	0.2	0.3	0.4	0.2	0.8	-0.2	15	1454.35sec
	0.4	0.3	0.2	0.1	0.2	0.8	-0.2	15	1768.26sec
	1.2	0.9	0.6	0.3	0.9	0.3	-0.7	391	1005.93sec
$n = 150$	0.1	0.2	0.3	0.4	0.2	0.8	-0.2	15	3271.54sec
	0.4	0.3	0.2	0.1	0.2	0.8	-0.2	15	3059.86sec
	1.2	0.9	0.6	0.3	1.5	0.2	-0.8	1238	1892.67sec
$n = 200$	0.1	0.2	0.3	0.4	0.2	0.8	-0.2	15	2757.2sec
	0.4	0.3	0.2	0.1	0.2	0.8	-0.2	15	4172.16sec
	1.2	0.9	0.6	0.3	0.3	0.7	-0.3	1086	1569.99sec

Πίνακας (**Εφαρμογή** 2)

Επίλυση του $\mathrm{Ax} = \mathrm{b}$ ESOR									
Διάσταση A	**Παράμετροι** (με δική σας επιλογή)				**Βέλτιστη τιμή**	**Βέλτιστη τιμή**	**Φασματική ακτίνα**	**Αριθμός επαναλήψεων**	**Χρόνος εκτέλεσης**
	α	β	γ	δ	τ_b	ω_b	$\rho(G(\tau_b,\omega_b))$	*itcount*	*cputime*
$n = 10^2$	0.1	0.1	0.1	0.1	0.1	1.3	0.3	13	558.091sec
	0.2	0.1	0.2	0.3	0.1	1.2	0.2	12	551.756sec
	0.3	0.2	-0.2	-0.3	0.3	0.5	-0.5	12	521.642sec
$n = 150$	0.1	0.1	0.1	0.1	0.1	1.3	0.3	13	1167.45sec
	0.2	0.1	0.2	0.3	0.7	0.3	-0.7	12	1075.08sec
	0.3	0.2	-0.2	-0.3	0.3	0.5	-0.5	12	991.67sec
$n = 200$	0.1	0.1	0.1	0.1	0.1	1.3	0.3	13	1912.48sec
	0.2	0.1	0.2	0.3	0.7	0.3	-0.7	12	1742.23sec
	0.3	0.2	-0.2	-0.3	0.3	0.5	-0.5	12	1633.82sec

Πίνακας (**Εφαρμογή** 3)

Επίλυση του $\mathrm{Ax} = \mathrm{b}$ ESOR									
Διάσταση A	**Παράμετροι** (με τη χρήση της rand())				**Βέλτιστη τιμή**	**Βέλτιστη τιμή**	**Φασματική ακτίνα**	**Αριθμός επαναλήψεων**	**Χρόνος εκτέλεσης**
	α	β	γ	δ	τ_b	ω_b	$\rho(G(\tau_b,\omega_b))$	*itcount*	*cputime*
$n = 10^2$	0.68891	0.372762	0.175105	0.293511	0.4	0.5	-0.5	24	468.817sec
	0.805827	-0.0733959	0.191226	0.859422	0.5	0.4	-0.6	20	443.309sec
	0.734158	0.157254	-0.0394286	0.325239	1.1	0.2	-0.8	16	389.843sec
$n = 150$	0.68891	0.372762	0.175105	0.293511	0.4	0.5	-0.5	24	934.499sec
	0.805827	-0.0733959	0.191226	0.859422	0.5	0.4	-0.6	20	887.956sec
	0.734158	0.157254	-0.0394286	0.325239	1.1	0.2	-0.8	16	779.465sec
$n = 200$	0.68891	0.372762	0.175105	0.293511	0.4	0.5	-0.5	24	1584.67sec
	0.805827	-0.0733959	0.191226	0.859422	0.5	0.4	-0.6	20	2553.8sec
	0.734158	0.157254	-0.0394286	0.325239	1.1	0.2	-0.8	16	1313sec

Για τον αλγόριθμο PSD:

Πίνακας (**Εφαρμογή** 1)

Επίλυση του $\mathrm{Ax} = \mathrm{b}$ PSD									
Διάσταση A	**Παράμετροι**				**Βέλτιστη τιμή**	**Βέλτιστη τιμή**	**Φασματική ακτίνα**	**Αριθμός επαναλήψεων**	**Χρόνος εκτέλεσης**
	α	β	γ	δ	τ_b	ω_b	$\rho(G(\tau_b,\omega_b))$	*itcount*	*cputime*
$n = 10^2$	0.1	0.2	0.3	0.4	1.3	0.2	-0.8	10	0.60797sec
	0.4	0.3	0.2	0.1	1.6	0.1	-0.9	10	0.577188sec
	1.2	0.9	0.6	0.3	0.6	0.3	-0.7	44	2.42552sec
$n = 1000$	0.1	0.2	0.3	0.4	0.9	0.1	-0.9	10	83.9531sec
	0.4	0.3	0.2	0.1	0.7	0.5	-0.5	10	75.0539sec
	1.2	0.9	0.6	0.3	1.3	1.3	0.3	43	303.429sec
$n = 3000$	0.1	0.2	0.3	0.4	0.9	0.7	-0.3	10	683.901sec
	0.4	0.3	0.2	0.1	1.4	0.6	-0.4	10	675.8sec
	1.2	0.9	0.6	0.3	0.5	0.4	-0.6	42	2885.78sec

Πίνακας (**Εφαρμογή** 2)

Επίλυση του $\mathrm{Ax} = \mathrm{b}$ PSD

Διάσταση A	**Παράμετροι** (με δική σας επιλογή) α	β	γ	δ	**Βέλτιστη τιμή** τ_b	**Βέλτιστη τιμή** ω_b	**Φασματική ακτίνα** $\rho(G(\tau_b,\omega_b))$	**Αριθμός επαναλήψεων** $itcount$	**Χρόνος εκτέλεσης** $cputime$
$n = 10^2$	0.1	0.1	0.1	0.1	0.4	0.4	-0.6	6	0.52795sec
	0.2	0.1	0.2	0.3	1.6	0.4	-0.6	9	0.810754sec
	0.3	0.2	-0.2	-0.3	1.5	0.1	-0.9	10	0.907285sec
$n = 10^3$	0.1	0.1	0.1	0.1	0.3	0.1	-0.9	6	69.6974sec
	0.2	0.1	0.2	0.3	0.8	1.7	-0.2	9	95.018sec
	0.3	0.2	-0.2	-0.3	0.3	0.3	-0.7	10	107.925sec
$n = 3000$	0.1	0.1	0.1	0.1	0.4	0.3	-0.7	6	586.549sec
	0.2	0.1	0.2	0.3	1.6	0.4	-0.6	9	849.36sec
	0.3	0.2	-0.2	-0.3	1.2	0.7	-0.3	10	936.226sec

Πίνακας (**Εφαρμογή** 3)

Επίλυση του $\mathrm{Ax} = \mathrm{b}$ PSD

Διάσταση A	**Παράμετροι** (με τη χρήση της rand()) α	β	γ	δ	**Βέλτιστη τιμή** τ_b	**Βέλτιστη τιμή** ω_b	**Φασματική ακτίνα** $\rho(G(\tau_b,\omega_b))$	**Αριθμός επαναλήψεων** $itcount$	**Χρόνος εκτέλεσης** $cputime$
$n = 10^2$	0.68891	0.372762	0.175105	0.293511	1.1	0.8	-0.2	14	1.27823sec
	0.805827	-0.0733959	0.191226	0.859422	1.5	1.2	0.2	19	1.71832sec
	0.734158	0.157254	-0.0394286	0.325239	1.7	0.4	-0.6	13	0.802513sec
$n = 10^3$	0.68891	0.372762	0.175105	0.293511	1.4	0.2	-0.8	14	163.9sec
	0.805827	-0.0733959	0.191226	0.859422	1.8	1.4	0.4	18	195.935sec
	0.734158	0.157254	-0.0394286	0.325239	0.9	0.1	-0.9	18	153.153sec
$n = 3000$	0.68891	0.372762	0.175105	0.293511	1.4	1.4	0.4	14	1384.81sec
	0.805827	-0.0733959	0.191226	0.859422	1.6	0.8	-0.2	18	1601.35sec
	0.734158	0.157254	-0.0394286	0.325239	0.7	0.2	-0.8	14	1235.54sec

Για την ESOR μέθοδο παρατηρούμε όσο αυξάνονται τα α,β,γ,δ η φασματική ακτίνα μικραίνει, η βέλτιστη τιμή του ω μεγαλώνει και η βέλτιστη τιμή του τ μικραίνει.

Παρατηρούμε και για τους δύο αλγορίθμους ότι όσο αυξάνεται ο όγκος των δεδομένων αυξάνεται και ο χρόνος που χρειάζεται ο αλγόριθμος για να τερματίσει. Όσο αυξάνεται το α ή το β ή το γ ή το δ αυξάνεται και ο αριθμός των επαναλήψεων που εκτελεί ο αλγόριθμος. Αυτό σημαίνει ότι χρειάζεται περισσότερο χρόνο για να συγκλίνει.

Επίσης, σύμφωνα με τους παραπάνω πίνακες παρατηρούμε ότι ο αλγόριθμος ESOR χρειάζεται πολύ περισσότερο χρόνο σε σχέση με τον PSD. Ο ESOR επίσης απαιτεί και μεγαλύτερο αριθμό επαναλήψεων για να συγκλίνει.