

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΗΥ200: ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ
ΦΛΩΡΟΥ ΧΡΥΣΟΥΛΑ ΑΕΜ:385

ΕΡΓΑΣΙΑ 1: Αριθμητική Γραμμική Άλγεβρα.

(Ημερομηνία Παράδοσης: Κυριακή 10 Απριλίου 2005, (Ώρα: 23:55))

(Η παράδοση της εργασίας θα γίνει μέσα από την σελίδα του μαθήματος στο eClass.)

ΠΙΝΑΚΑΣ ΑΣΚΗΣΕΩΝ 1,2

n_x	n_y	c_1	c_2	c	χρόνος	σφάλμα
3	10	0	0	0	0.01	$8.0005e-015$
10	3	0	0	0	0.02	$1.6852e-014$
5	5	0	0	0	0	$3.7321e-015$
5	5	100	0	0	0.028	$2.6899e-015$
5	5	0	100	0	0.045	$3.2501e-015$
5	5	0	0	100	0	$6.3273e-015$
10	10	0	0	0	0.018	$2.6232e-014$
20	20	0	0	0	0.22	$2.599e-013$
30	30	0	0	0	0.7513	$1.0038e-012$
40	40	0	0	0	3.356	$2.6837e-012$
50	50	0	0	0	14.31	$7.0978e-012$
10	10	0	0	100	0.001	$4.9728e-014$
20	20	0	0	100	0.185	$2.5738e-013$
30	30	0	0	100	0.681	$1.0141e-012$
40	40	0	0	100	2.98	$2.8267e-012$
50	50	0	0	100	11.14	$7.4974e-012$

Πίνακας 1. Αποτελέσματα με χρήση της `lu`.

ΠΑΡΑΤΗΡΗΣΕΙΣ:

ΑΣΚΗΣΗ 1

Το σφάλμα είναι της τάξης $e-015$, δηλαδή είναι αρκετά μικρό, επομένως η λύση μας είναι αρκετά ακριβής. Επειδή η `lu` είναι άμεση μέθοδος και όχι επαναληπτική, το σφάλμα οφείλεται μόνο στην πεπερασμένη αριθμητική του υπολογιστή και γι' αυτό είναι μικρότερο σε σχέση με το σφάλμα των επαναληπτικών μεθόδων.

ΑΣΚΗΣΗ 2

Παρατηρούμε ότι αυξάνοντας τα n_x και n_y ο χρόνος εκτέλεσης αυξάνεται. Αυτό γίνεται επειδή η `lu.run` καλεί την συνάρτηση `elliptic` η οποία περιέχει βρόγχους που εξαρτώνται από τιμές των n_x και n_y . Άρα αυξάνοντας τα n_x και n_y αυξάνεται το πλήθος των επαναλήψεων των βρόγχων, άρα και ο χρόνος ολοκλήρωσής τους. Βλέπουμε, ότι όσο αυξάνονται τα n_x και n_y το σφάλμα αυξάνεται και για $c=0$ και για $c=100$ (αφού το c επηρεάζει μόνο τη γραφική παράσταση). Επομένως η λύση μας γίνεται όλο και λιγότερο ακριβής.

ΑΣΚΗΣΗ 3

Η `luinc.run` χρησιμοποιεί την συνάρτηση `sparse` η οποία μας δίνει τα στοιχεία του αραιού πίνακα a καθώς και την θέση αυτών στον αραιό πίνακα. Επομένως, ο χρόνος εκτέλεσης της `luinc.run` θα είναι μικρότερος από τον χρόνο εκτέλεσης της `lu.run`. Αφού η τελευταία χρησιμοποιεί τον αραιό πίνακα και εξετάζει όλα τα στοιχεία του, είναι λογικό να είναι πιο χρονοβόρα από την `luinc.run`.

ΑΣΚΗΣΗ 4

Παρατηρούμε ότι όταν $c=100$ ο χρόνος εκτέλεσης είναι μικρότερος σε σχέση με τον χρόνο εκτέλεσης όταν $c=0$. Όταν $c=0$ γίνονται όλες οι επαναλήψεις επειδή το σφάλμα δεν γίνεται ποτέ μικρότερο από το $\text{eps}=0.5e-5$. Αντιθέτως για $c=100$ δεν γίνονται όλες οι επαναλήψεις επειδή το σφάλμα γίνεται μικρότερο του eps . Δηλαδή ο βρόγχος `while(iter<nx*ny error>eps)` κάπου διακόπτεται εξαιτίας του ότι $\text{error}<\text{eps}$. Επομένως για $c=0$ το σφάλμα είναι μεγαλύτερο απ' ό,τι για $c=100$ και άρα η ακρίβεια της λύσης είναι καλύτερη για $c=100$.

ΑΣΚΗΣΗ 5

Εδώ παρατηρούμε ότι και για $c=0$ και για $c=100$ δεν γίνονται όλες οι επαναλήψεις επειδή το σφάλμα γίνεται μικρότερο του eps . Εκτός από την περίπτωση που $n_x=n_y=10$ και $c=0$ όπου γίνονται όλες οι επαναλήψεις. Συγκεκριμένα, στην G-S έχουμε λιγότερες επαναλήψεις (3 for) από την Jacobi (4 for).

Ακόμη παρατηρούμε ότι είναι πιο ακριβής από την Jacobi. Το αποτέλεσμα κάθε επανάληψης `xnew` εξαρτάται από όλα τα προηγούμενα `xold` και ο χρόνος εκτέλεσης της είναι μεγαλύτερος από τον χρόνο εκτέλεσης της Jacobi, αφού το αποτέλεσμα `xnew` της Jacobi εξαρτάται μόνο από το προηγούμενο `xold`.

ΑΣΚΗΣΗ 6

Παρατηρούμε ότι η CG κάνει λιγότερες επαναλήψεις από τις δύο προηγούμενες μεθόδους και γι' αυτό το λόγο το σφάλμα της είναι μικρότερο από το σφάλμα των άλλων δύο, άρα η λύση της είναι πιο ακριβής. Τέλος, ο χρόνος εκτέλεσης της είναι πολύ πιο μικρός από το χρόνο εκτέλεσης των προηγούμενων δύο μεθόδων.

ΠΙΝΑΚΑΣ ΑΣΚΗΣΕΩΝ 3,4,5

	$n_x = n_y$	c	χρόνος	επαναλήψεις	σφάλμα	c	χρόνος	επαναλήψεις	σφάλμα
<i>luinc</i>	10	0	0.211		$2.6472e - 014$	100	0		$4.6573e - 014$
	20	0	0		$2.8339e - 013$	100	0.018		$2.7947e - 013$
	30	0	0.23		$1.0538e - 012$	100	0.037		$1.1591e - 012$
	40	0	0.09		$2.851e - 012$	100	0.078		$3.0609e - 012$
	50	0	0.34		$7.4437e - 012$	100	0.38		$7.6968e - 012$
<i>Jacobi</i>	10	0	0.014	100	$2.4734e - 004$	100	0.013	43	$4.5931e - 006$
	20	0	0.256	400	$8.8342e - 005$	100	0.097	139	$4.7231e - 006$
	30	0	3.48	900	$5.1935e - 005$	100	2.3	285	$4.9409e - 006$
	40	0	18.1	1600	$3.6468e - 005$	100	6.2	480	$4.9761e - 006$
	50	0	74.87	2500	$2.8015e - 005$	100	20.1	722	$4.9764e - 006$
<i>G - S</i>	10	0	0.56	100	$8.4316e - 006$	100	0.48	25	$4.3281e - 006$
	20	0	8.234	360	$4.9394e - 006$	100	6.71	77	$4.3936e - 006$
	30	0	45.16	746	$4.9818e - 006$	100	27.2	156	$4.7805e - 006$
	40	0	208.1	1257	$4.9941e - 006$	100	87.9	262	$4.8521e - 006$
	50	0	498.97	1888	$4.9845e - 006$	100	224	393	$4.9679e - 006$
<i>CG</i>	10	0	0	14	$2.544e - 007$	100	0	12	$9.5757e - 007$
	20	0	0.067	29	$2.8347e - 006$	100	0.031	24	$2.7685e - 007$
	30	0	0.56	44	$3.685e - 006$	100	0.51	36	$4.4713e - 007$
	40	0	1.78	59	$3.9598e - 006$	100	1.24	49	$3.7638e - 006$
	50	0	4.78	75	$4.2605e - 006$	100	3.8	62	$3.7857e - 006$

Πίνακας 2. Αποτελέσματα με χρήση λογισμικού για αραιούς πίνακες και επαναληπτικές μεθόδους.