

Όνομα : Marie-Aurelie Nef

Username : manef

ΑΕΜ : 370

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΗΥ200: ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ

ΕΡΓΑΣΙΑ 1: Αριθμητική Γραμμική Άλγεβρα.

ΑΣΚΗΣΗ 1.

α) Καλέστε την `lu_run` για να κάνετε παραγοντοποίηση με την `lu`, και την `\` για να λύσετε τα τριγωνικά συστήματα που προκύπτουν, αν το αρχικό προέρχεται από $n_x = 3, n_y = 10, c_1 = 0, c_2 = 0, c = 0$.

β) Όμοια για $n_x = 10, n_y = 3, c_1 = 0, c_2 = 0, c = 0$.

γ) Όμοια για $n_x = 5, n_y = 5, c_1 = 0, c_2 = 0, c = 0$.

δ) Όμοια για $n_x = 5, n_y = 5, c_1 = 100, c_2 = 0, c = 0$.

ε) Όμοια για $n_x = 5, n_y = 5, c_1 = 0, c_2 = 100, c = 0$.

στ) Όμοια για $n_x = 5, n_y = 5, c_1 = 0, c_2 = 0, c = 100$.

Τι παρατηρείτε ως προς την ακρίβεια της λύσης σας;

n_x	n_y	c_1	c_2	c	χρόνος	σφάλμα
3	10	0	0	0	0	$8,000542 \cdot 10^{-15}$
10	3	0	0	0	0	$1,6852 \cdot 10^{-14}$
5	5	0	0	0	0	$3,732067 \cdot 10^{-15}$
5	5	100	0	0	0	$2,689858 \cdot 10^{-15}$
5	5	0	100	0	0	$3,250129 \cdot 10^{-15}$
5	5	0	0	100	0	$6,327297 \cdot 10^{-15}$

Πίνακας 1. Αποτελέσματα με χρήση της `lu`.

Όταν το n_y είναι μεγαλύτερο του n_x , έχουμε πιο μικρό σφάλμα.

Αντίστοιχα, όταν το n_y είναι μικρότερο από το n_x , παρατηρούμε μια αρκετά μεγάλη αύξηση του σφάλματος.

Όταν έχουμε 100 στις σταθερές c_1, c_2 και c , το σφάλμα αυξάνει αντίστοιχα.

ΑΣΚΗΣΗ 2.

α) Όμοια για $n_x = 10, n_y = 10, c_1 = 0, c_2 = 0$ και $c = 0$.

β) Όμοια για $n_x = 20, n_y = 20, c_1 = 0, c_2 = 0, c = 0$ και $c = 100$.

γ) Όμοια για $n_x = 30, n_y = 30, c_1 = 0, c_2 = 0, c = 0$ και $c = 100$.

δ) Όμοια για $n_x = 40, n_y = 40, c_1 = 0, c_2 = 0, c = 0$ και $c = 100$.

ε) Όμοια για $n_x = 50, n_y = 50, c_1 = 0, c_2 = 0, c = 0$ και $c = 100$.

Τι παρατηρείτε ως προς την ακρίβεια της λύσης σας και ως προς το χρόνο εκτέλεσης;

n_x	n_y	c_1	c_2	c	χρόνος	σφάλμα
10	10	0	0	0	0	$2,623176 \cdot 10^{-14}$
20	20	0	0	0	0,032	$2,599005 \cdot 10^{-13}$
30	30	0	0	0	0,328	$1,0038113 \cdot 10^{-12}$
40	40	0	0	0	1,719	$2,683659 \cdot 10^{-12}$
50	50	0	0	0	6,5	$7,097778 \cdot 10^{-12}$
10	10	0	0	100	0,016	$4,972808 \cdot 10^{-14}$
20	20	0	0	100	0,031	$2,573832 \cdot 10^{-13}$
30	30	0	0	100	0,344	$1,014127 \cdot 10^{-12}$
40	40	0	0	100	1,703	$2,826693 \cdot 10^{-12}$
50	50	0	0	100	6,407	$7,497432 \cdot 10^{-12}$

Πίνακας 2. Αποτελέσματα με χρήση της `lu`. (Συνέχεια)

Όταν αυξάνουν τα ζευγάρια n_x και n_y , αυξάνουν το σφάλμα και ο χρόνος εκτέλεσης.

Με την σταθερά $c = 100$, δεν υπάρχει σημαντική διαφορά στο χρόνο εκτέλεσης που είχαμε με $c = 0$, όμως το σφάλμα αυξάνει (με μία σημαντική διαφορά για $n_x = 10$ και $n_y = 10$)

ΑΣΚΗΣΗ 3.

Δοκιμάστε να λύσετε τα τελευταία 5 συστήματα, της άσκησης 2, χρησιμοποιώντας την βασική ιδιότητα του A , ότι είναι αραιός. Μελετήστε την συνάρτηση `sparse` του Matlab και χρησιμοποιείστε την για μετατρέψετε και να αποθηκεύσετε τον A σε αραιή μορφή. Επαναλάβετε τις εντολές για τη λύση του συστήματος, μόνο που τώρα θα χρησιμοποιήσετε την `luinc` αντί της `lu` μέσα από την `luinc.run`. Τι παρατηρείτε ως προς το χρόνο εκτέλεσης;

	$n_x = n_y$	c	χρόνος	επαν.	σφάλμα	c	χρόνος	επαν.	σφάλμα
<i>luinc</i>	10	0	0	—	$3,423773 \cdot 10^{-5}$	100	0,031	—	$4,494182 \cdot 10^{-4}$
	20	0	0	—	$2,221416 \cdot 10^{-3}$	100	0,016	—	$7,352285 \cdot 10^{-3}$
	30	0	0,015	—	$1,442299 \cdot 10^{-2}$	100	0,015	—	$3,470693 \cdot 10^{-2}$
	40	0	0,063	—	$5,676341 \cdot 10^{-2}$	100	0,063	—	$1,01805 \cdot 10^{-1}$
	50	0	0,141	—	$1,660909 \cdot 10^{-1}$	100	0,125	—	$2,439775 \cdot 10^{-1}$

Πίνακας 3. Αποτελέσματα με χρήση της *luinc*.

Υπάρχει, όσο μεγαλώνουν τα n_x και n_y , αύξηση στο χρόνο εκτέλεσης όσο μεγαλώνουν τα n_x και n_y αλλά όχι τόσο αξιοσημείωτη.

Επίσης παρατηρούμε μία μεγάλη αύξηση του σφάλματος που είναι και πολύ πιο μεγάλο απ'αυτό που είχαμε με την *lu* μέθοδο.

ΑΣΚΗΣΗ 4.

Χρησιμοποιώντας πάλι την ιδιότητα του A , ότι είναι αραιός, καλέστε τη *jacobi*, η οποία θα υλοποιεί την επαναληπτική μέθοδο Jacobi, μέσα από την *jacobi-run*.

Δοκιμάστε να λύσετε τα τελευταία 5 συστήματα από την άσκηση 2. Τι παρατηρείτε ως προς την ακρίβεια της λύσης σας και ως προς το χρόνο εκτέλεσης και πλήθος επαναλήψεων;

	$n_x = n_y$	c	χρόνος	επαν.	σφάλμα	c	χρόνος	επαν.	σφάλμα
<i>Jacobi</i>	10	0	0,015	100	$2,473362 \cdot 10^{-4}$	100	0,016	43	$4,953114 \cdot 10^{-6}$
	20	0	0,828	400	$8,834175 \cdot 10^{-5}$	100	0,328	139	$4,723127 \cdot 10^{-6}$
	30	0	7,922	900	$5,193455 \cdot 10^{-5}$	100	2,875	285	$4,940874 \cdot 10^{-6}$
	40	0	50,578	1600	$3,646808 \cdot 10^{-5}$	100	15,75	480	$4,976084 \cdot 10^{-6}$
	50	0	186,203	2500	$2,801544 \cdot 10^{-5}$	100	59,641	722	$4,976429 \cdot 10^{-6}$

Πίνακας 4. Αποτελέσματα με χρήση της *Jacobi*.

Με $c = 0$, αύξοντα αριθμό επαναλήψεων, βλέπουμε ότι το σφάλμα μικραίνει. Παρατηρούμε επίσης ότι οι επαναλήψεις σταματούν εκεί που τις ορίσαμε αν και το σφάλμα δεν μας ικανοποιεί.

Με $c = 100$, ο χρόνος εκτέλεσης και οι επαναλήψεις μειώθηκαν κατά πολύ και έχουμε σχεδόν σταθερό σφάλμα.

ΑΣΚΗΣΗ 5.

Τροποποιήστε την *jacobi.m* ώστε να υλοποιήσετε την επαναληπτική μέθοδο Gauss-Seidel μέσα στην

gs. m την οποία θα καλέσετε κλασσικά μέσα από την gs_run.

Δοκιμάστε να λύσετε τα ίδια τελευταία 5 συστήματα. Τι παρατηρείτε ως προς την ακρίβεια της λύσης σας και ως προς το χρόνο εκτέλεσης και πλήθος επαναλήψεων;

	$n_x = n_y$	c	χρόνος	επαν.	σφάλμα	c	χρόνος	επαν.	σφάλμα
$G - S$	10	0	0,61	100	$8,431617 \cdot 10^{-6}$	100	0,046	25	$4,328123 \cdot 10^{-6}$
	20	0	1,546	360	$4,939364 \cdot 10^{-6}$	100	0,36	77	$4,393552 \cdot 10^{-6}$
	30	0	16,204	746	$4,9818 \cdot 10^{-6}$	100	3,515	156	$4,780462 \cdot 10^{-6}$
	40	0	85,032	1257	$4,994124 \cdot 10^{-6}$	100	20,813	262	$4,852141 \cdot 10^{-6}$
	50	0	285,516	1888	$4,984559 \cdot 10^{-6}$	100	54,937	393	$4,967869 \cdot 10^{-6}$

Πίνακας 5. Αποτελέσματα με χρήση της Gauss-Seidel.

Με $c = 0$, για $n_x = 10$ και $n_y = 10$, φτάνει στο όριο των επαναλήψεων και έτσι έχουμε μεγαλύτερο σφάλμα. Για τις άλλες περιπτώσεις, όσο αυξάνουν τα n_x και n_y , αυξάνουν και ο χρόνος εκτέλεσης και οι επαναλήψεις αλλά το σφάλμα παραμένει σχετικά σταθερό.

Με $c = 100$, ο χρόνος εκτέλεσης και οι επαναλήψεις μειώθηκαν κατά πολύ και έχουμε σχεδόν σταθερό σφάλμα.

ΑΣΚΗΣΗ 6.

Καλέστε την cg. m μέσα από την cg_run για να χρησιμοποιήσετε την μέθοδο Συζηγών Κλίσεων (Conjugate Gradients) στην επίλυση των γραμμικών συστημάτων.

Δοκιμάστε να λύσετε τα ίδια τελευταία 5 συστήματα. Τι παρατηρείτε ως προς την ακρίβεια της λύσης σας και ως προς το χρόνο εκτέλεσης και πλήθος επαναλήψεων;

	$n_x = n_y$	c	χρόνος	επαν.	σφάλμα	c	χρόνος	επαν.	σφάλμα
CG	10	0	0,235	14	$2,544042 \cdot 10^{-7}$	100	0	12	$9,575718 \cdot 10^{-7}$
	20	0	0,125	29	$2,834746 \cdot 10^{-6}$	100	0,094	24	$2,768481 \cdot 10^{-6}$
	30	0	0,75	44	$3,684995 \cdot 10^{-6}$	100	0,61	36	$4,471294 \cdot 10^{-6}$
	40	0	3,765	59	$3,959815 \cdot 10^{-6}$	100	3,141	49	$3,763833 \cdot 10^{-6}$
	50	0	10	75	$4,260508 \cdot 10^{-6}$	100	8,437	62	$3,785689 \cdot 10^{-6}$

Πίνακας 6. Αποτελέσματα με χρήση της Conjugate Gradients.

Με $c = 0$, όσο αυξάνουν τα ζευγάρια n_x και n_y , ο χρόνος και οι επαναλήψεις αυξάνουν αντίστοιχα αλλά πολύ πιο αργά από τις Jacobi και Gauss-Seidel, το σφάλμα αυξάνει επίσης.

Με $c = 100$, παρατηρούμε σχεδόν τα ίδια πράγματα.

Γενικό Σχόλιο

- Για τις μη επαναληπτικές μεθόδους:

Αν θέλουμε να έχουμε το μικρότερο σφάλμα και σχετικά μικρό χρόνο εκτέλεσης, θα προτιμήσουμε την lu μέθοδο. Αν όμως θέλουμε πολύ μικρό χρόνο εκτέλεσης και όχι τόσο καλή ακρίβεια στην λύση μας, θα επιλέξουμε την luinc μέθοδο που δουλεύει πιο γρήγορα εφόσον δεν κάνει τόσες πράξεις.

- Για τις επαναληπτικές μεθόδους:

Παρατηρούμε ότι η πιο γρήγορη από τις τρεις που δοκιμάσαμε είναι η μέθοδος Συζηγών Κλίσεων εφόσον ο αριθμός επαναλήψεων είναι πολύ πιο μικρός. Η μέθοδος αυτή μας δίνει επίσης μία καλή ακρίβεια (γενικά το σφάλμα είναι και λίγο μικρότερο από το σφάλμα που είχαμε με τις Jacobi και Gauss-Seidel).