Ρεϊζόπουλος Μάριος ΑΕΜ: 5782
Τερζής Δημήτριος ΑΕΜ:6101

4η Εργασία Υπολογιστική Ρευστομηχανική

Εισαγωγή

Αντικείμενο της εργασίας αποτελούσε ο σχεδιασμός ενός μοντέλου υπολογιστικής ρευστομηχανικής, για την μελέτη της ασυμπίεστης ροής (incompressible flow) σε έναν 2D αγωγό όπου η εισαγωγή το ρευστού γίνεται οριζόντια (x διεύθυνση) και έπειτα έχουμε αλλαγή φοράς της φοράς του αγωγού κατά 90° και έξοδο στην y διεύθυνση. Κληθήκαμε να δημιουργήσουμε μέσω της Python, ένα πρόγραμμα το οποίο, με χρήση της μεθόδου διόρθωσης πίεσης SIMPLE, θα μπορεί να επιλύει την ροή αυτή.

Περιγραφή μαθηματικού μοντέλου

Η ροή μας, ως ασυμπίεστη αλλά μη ιδανική (ιξώδες παρών), περιγράφεται με ακρίβεια από τις συναρτήσεις Navier-Stokes, οι οποίες περιγράφουν την ορμή της ροής στην χ και y κατεύθυνση και έχουν ως εξής:

U-Momentum

$$\frac{\partial(\rho u u)}{\partial x}+\frac{\partial(\rho u v)}{\partial y}=-\frac{\partial p}{\partial x}+\left[\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right)\right]$$

V-Momentum

$$\frac{\partial(\rho u v)}{\partial x}+\frac{\partial(\rho v v)}{\partial y}=-\frac{\partial p}{\partial y}+\left[\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)\right]$$

Επίσης, για να έχουμε μια λύση που ανταποκρίνεται στην πραγματικότητα, χρειάζεται να επαληθεύεται σε κάθε σημείο του πλέγματος μας η εξίσωση της συνέχειας η οποία έχει ως εξής:

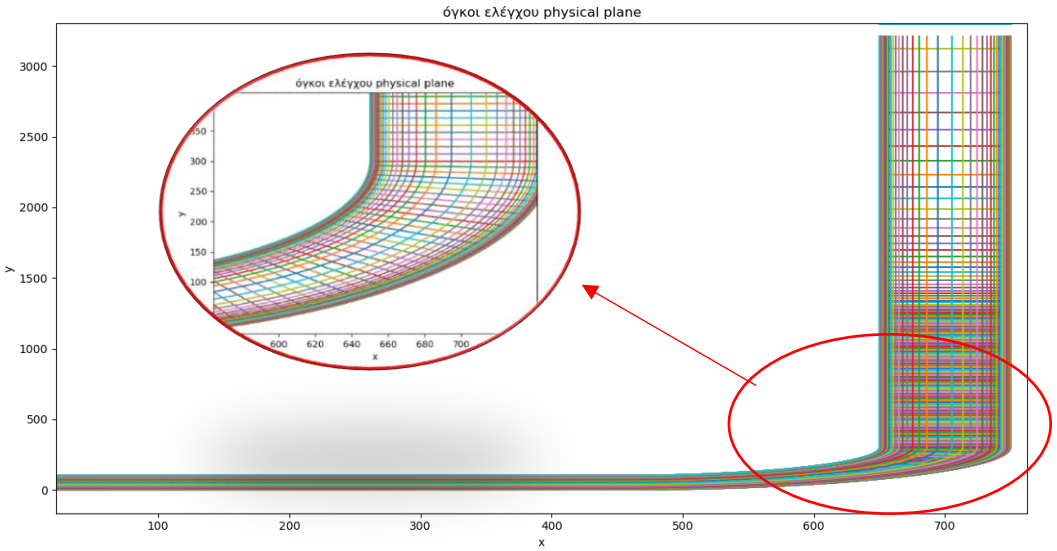
Continuity

$$\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}=0$$

Το πρόβλημα

Η γεωμετρία και άρα και το πλέγμα προς μελέτη είναι το εξής:

Όπως μπορεί να παρατηρήσει κανείς, τόσο λόγω της τεχνικής της πύκνωσης πλέγματος, αλλά και λόγω της μη-ευθύγραμμης φύσης της γεωμετρίας θα παρουσιαστούν προβλήματα όταν θα κληθούμε να διακριτοποιήσουμε τις παραγώγους των παραπάνω εξισώσεων.



Ρεϊζόπουλος Μάριος ΑΕΜ: 5782
Τερζής Δημήτριος ΑΕΜ:6101

Η λύση

Για να επιλύσουμε λοιπόν το σύστημα, θα χρησιμοποιήσουμε την τεχνική του μετασχηματισμένου πλέγματος, όπου θα «στρεβλώσουμε » το πλέγμα μας, ώστε κάθε φορά που παραγωγίζουμε, το διάνυσμα της εκάστοτε ταχύτητας να είναι κάθετο στην επιφάνεια ελέγχου στην οποία υπάγεται. Το μετασχηματισμένο αυτό πλέγμα επίσης, θα έχει την ιδιότητα όπου όλες οι αποστάσεις μεταξύ των σημείων, να είναι ίσες μεταξύ τους και ίσες με 1.

Η τεχνική αυτή, κάνει χρήση της Ιακωβιανής ορίζουσας J, ώστε να μεταφερθούμε πλέον από το (χ,y) επίπεδο, στο (ξ, η) επίπεδο.

Ο μετασχηματισμός έχει ως εξής:

$$\frac{\partial}{\partial x} = \frac{1}{J}\left[\left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial y}{\partial \eta}\right) - \left(\frac{\partial u}{\partial \eta}\right)\left(\frac{\partial y}{\partial \xi}\right)\right]$$

$$\frac{\partial}{\partial y} = \frac{1}{J}\left[\left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial x}{\partial \xi}\right) - \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial x}{\partial \eta}\right)\right]$$

Όπου για λόγους ευκολίας θα ονομάσουμε τις παραγώγους αυτές ανάλογα με την θέση τους στην Ιακωβιανή ορίζουσα

$$b22 = \frac{\partial y}{\partial \eta}, b21 = \frac{\partial y}{\partial \xi}, b11 = \frac{\partial x}{\partial \xi}, b12 = \frac{\partial x}{\partial \eta}$$

Η οποία έχει ως εξής:

$$J = \begin{vmatrix} b11 & b12 \\ b21 & b22 \end{vmatrix}$$

Οπότε τελικά, οι μετασχηματισμένες εξισώσεις ορμής και συνέχειας είναι:

U momentum:

$$\frac{\partial(\rho(u\,\frac{\partial y}{\partial \eta} - v\frac{\partial x}{\partial \eta})u)}{\partial \xi} + \frac{\partial(\rho(u\,\frac{\partial y}{\partial \eta} - v\frac{\partial x}{\partial \eta})u)}{\partial \eta} = -\quad \{\frac{\partial y}{\partial \eta}\frac{\partial p}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial p}{\partial \eta}\} + \frac{\partial}{\partial \xi}[\frac{\mu}{J}(q_1 u_\xi - q_2 u_\eta)] + \frac{\partial}{\partial \eta}[\frac{\mu}{J}(-q_2 u_\xi + q_3 u_\eta)]$$

V-momentum

$$\frac{\partial(\rho(u\,\frac{\partial y}{\partial \eta} - v\frac{\partial x}{\partial \eta})v)}{\partial \xi} + \frac{\partial\left(\rho(v\frac{\partial x}{\partial \xi} - u\frac{\partial y}{\partial \xi})v\right)}{\partial \eta}$$
$$= -\{\frac{\partial x}{\partial \xi}\frac{\partial p}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial p}{\partial \xi}\} + \frac{\partial}{\partial \xi}[\frac{\mu}{J}(q_1\frac{\partial v}{\partial \xi} - q_2\frac{\partial v}{\partial \eta})] + \frac{\partial}{\partial \eta}[\frac{\mu}{J}(-q_2\frac{\partial v}{\partial \xi} + q_3\frac{\partial v}{\partial \eta})]$$

Με

$$q_1 = x_\eta^2 + y_\eta^2$$
$$q_2 = x_\xi x_\eta + y_\xi y_\eta$$
$$q_3 = x_\xi^2 + y_\xi^2$$

Οι παραπάνω εξισώσεις, μπορούν να χωριστούν σε 3 μέρη: Σε αυτό της συναγωγής (αριστερό μέρος της εξίσωσης), σε αυτό της διάχυσης(παράγωγοι δεύτερης τάξης) και στον όρο πηγής (διαφορά πίεσης στον όγκο ελέγχου). Σύμφωνα και με το βιβλίο του Malalasekera, οι όροι αυτοί συμβολίζονται με F: όρος συναγωγής, D: όρος διάχυσης και S: όρος πηγής.

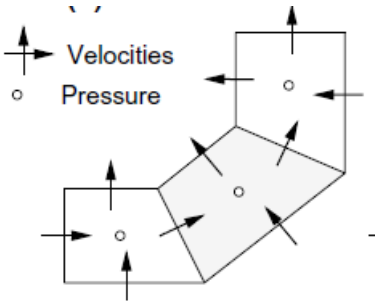
Ρεϊζόπουλος Μάριος ΑΕΜ: 5782
Τερζής Δημήτριος ΑΕΜ:6101

Η μέθοδος πεπερασμένων όγκων

Επόμενο βήμα στην επίλυση, είναι ο σχηματισμός όγκων ελέγχου, γύρω από τα κεντρικά σημεία του πλέγματος, ώστε να μπορούμε σε κάθε σημείο να ολοκληρώνουμε τις ποσότητες που μπαίνουν και βγαίνουν από τον όγκο. Η ολοκλήρωση του όγκου ελέγχου η οποία αποτελεί το βασικό στάδιο της μεθόδου πεπερασμένων όγκων και τη διακρίνει από όλες τις υπόλοιπες τεχνικές cfd, δίνει την παρακάτω μορφή στις εξισώσεις που διέπουν την ροή μας:

U momentum:

$$\left(\varrho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})u-\frac{\mu}{J}\left(q_1\frac{\partial u}{\partial \xi}-q_2\frac{\partial u}{\partial \eta}\right)+\frac{\partial y}{\partial \eta}p\right)_e$$
$$-\left(\rho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})u-\frac{\mu}{J}\left(q_1\frac{\partial u}{\partial \xi}-q_2\frac{\partial u}{\partial \eta}\right)+\frac{\partial y}{\partial \eta}p\right)_w$$
$$+\left(\rho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})u-\frac{\mu}{J}\left(-q_2\frac{\partial u}{\partial \xi}+q_3\frac{\partial u}{\partial \eta}\right)-\frac{\partial y}{\partial \xi}p\right)n$$
$$-(\rho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})u-\frac{\mu}{J}(-q_2\frac{\partial u}{\partial \xi}+q_3\frac{\partial u}{\partial \eta})-\frac{\partial y}{\partial \xi}p)_s=0$$



V-momentum

$$(\varrho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})v-\frac{\mu}{J}(q_1\frac{\partial v}{\partial \xi}-q_2\frac{\partial v}{\partial \eta})+\frac{\partial x}{\partial \eta}p)_e$$
$$-(\varrho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})v-\frac{\mu}{J}(q_1\frac{\partial v}{\partial \xi}-q_2\frac{\partial v}{\partial \eta})+\frac{\partial x}{\partial \eta}p)_v$$
$$+(\varrho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})v-\frac{\mu}{J}(-q_2\frac{\partial v}{\partial \xi}+q_3\frac{\partial v}{\partial \eta})-\frac{\partial x}{\partial \xi}p)_n$$
$$-(\varrho(u\frac{\partial y}{\partial \eta}-v\frac{\partial x}{\partial \eta})v-\frac{\mu}{J}(-q_2\frac{\partial v}{\partial \xi}+q_3\frac{\partial v}{\partial \eta})-\frac{\partial x}{\partial \xi}p)_s=0$$

Όπου e, w, s, n, είναι οι έδρες του εκάστοτε όγκου ελέγχου

Το μετατοπισμένο πλέγμα

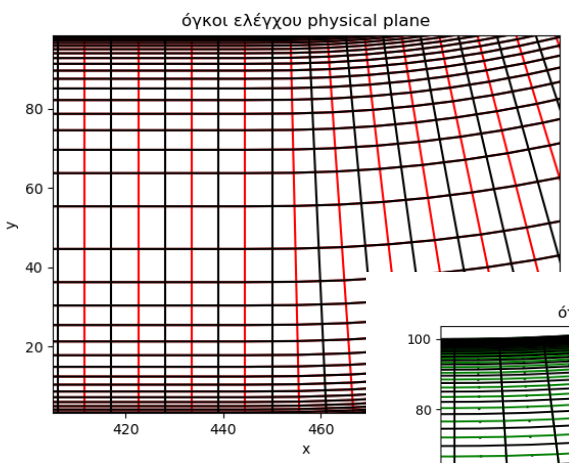
Αφού έχουμε μετασχηματίσει τις εξισώσεις μεταφοράς μας, σειρά έχει να βρούμε πού θα αποθηκεύονται οι τιμές της u και της v ταχύτητας, καθώς και της πίεσης p. Εάν αποθηκεύονταν στο ίδιο σημείο (κέντρο όγκων ελέγχου), τότε ένα πεδίο πίεσης υψηλής ανομοιομορφίας, θα είχε ως αποτέλεσμα να μηδενίσει πολλές φορές τις διακριτοποιημένες εξισώσεις ορμής και να επιφέρει μη αληθοφανή αποτελέσματα.

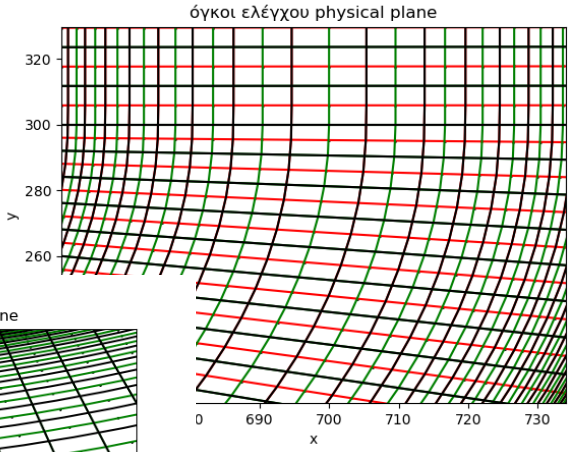
Για να αντιμετωπίσουμε το φαινόμενο αυτό, εφαρμόζουμε την τεχνική του μετατοπισμένου πλέγματος. Έτσι, θα ορίσουμε τους u-όγκους ελέγχου (κόκκινους), όπου επιλύουμε τις u εξισώσεις ορμής, τους v-όγκους ελέγχου (πράσινους) όπου επιλύουμε τις v εξισώσεις ορμής και τους p, κεντρικούς, όγκους ελέγχου (μαύροι) όπου αργότερα κάνουμε την διόρθωση της πίεσης και υπολογίζουμε την συνέχεια. όπως παρουσιάζονται και στις παρακάτω εικόνες, ενώ οι τιμές των ταχυτήτων, θα αποθηκεύονται στα αντίστοιχα κέντρα των μετατοπισμένων όγκων αυτών.

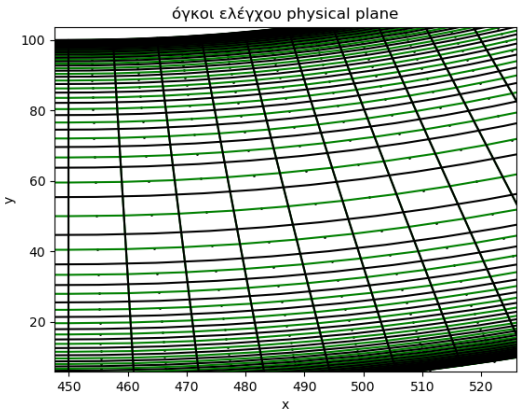
Τα νέα πλέγματα αυτά, χρειάζονται επίσης να μετασχηματιστούν στον (ξ, η) επίπεδο με την βοήθεια των αντίστοιχων metrics και των Ιακοβιανών οριζουσών τους.

Άρα λοιπόν, έχουμε πλέον 3 διαφορετικά πλέγματα με 3 διαφορετικές Ιακωβιανές ορίζουσες, και υπολογίζουμε τις εξισώσεις μεταφοράς αντίστοιχα στο καθένα

Ρεϊζόπουλος Μάριος ΑΕΜ: 5782
Τερζής Δημήτριος ΑΕΜ:6101







Διακριτοποίηση

Για να επιλύσουμε οπότε το πεδίο της ροής πρέπει να διακριτοποιήσουμε περεταίρω τις εξισώσεις της ορμής γύρω από κάθε έδρα των μετατοπισμένων u και v όγκων ελέγχου.

Η γενική μορφή του συστήματός μας προς επίλυση στην u διεύθυνση είναι:

$$a_p u_p = a_e u_e + a_w u_w + a_{n^u n} + a_s u_s + S_u$$

Όπου ο δείκτης p, υποδεικνύει τον προς μελέτη κέντρο του όγκου ελέγχου, ενώ οι υπόλοιποι υποδεικνύουν τις αντίστοιχες έδρες του.

Συγκεκριμένα, οι συντελεστές α, υπολογίζονται μέσω του αθροίσματος των όρων διάχυσης και συναγωγής των αντίστοιχων εδρών:

$$a_e = -F|_e + D|_e$$
$$a_w = \mathrm{F}|_w + \mathrm{D}|_w$$
$$a_n = -F|_n + D\mathrm{I}_n$$
$$a_s = \mathrm{F}|_s + \mathrm{D}|_s$$

Για να διακριτοποιήσει κανείς τους όρους διάχυσης και συναγωγής, χρειάζεται να επιλέξει σχήμα διακριτοποίησης και να γνωρίζει τις θέσεις που είναι αποθηκευμένες οι τιμές της αντίστοιχης ταχύτητας.

Εδώ χρησιμοποιήθηκε η μέθοδος της κεντρικής διακριτοποίησης,

$$F_w = \frac{1}{2}\rho[\left(\frac{\partial y}{\partial \eta} * u_p - \frac{\partial x}{\partial \eta} * v_p\right) - \left(\frac{\partial y}{\partial \eta} * u_w - \frac{\partial x}{\partial \eta} * v_w\right)]$$

Όπου $v_p = \frac{v_N + v_{NW} + v_W + v_p}{4}$, $v_w = \frac{v_N + v_{NW} + v_N + v_p}{4}$

$$D_w = \frac{\mu}{J} * q_1 + \frac{\mu}{J} * q_2 \frac{\partial u}{\partial \eta}| = \frac{\mu}{J} * [q_1 + q_2(\frac{\frac{u_{NW} + u_N + u_P + u_W}{4} \_ \frac{u_W + u_P + u_{SW} + u_S}{4}}{2 * \Delta\xi})$$

$$F_e = \frac{1}{2}\rho[\left(\frac{\partial y}{\partial \eta} * u_p - \frac{\partial y}{\partial \eta} * v_p\right) - \left(\frac{\partial y}{\partial \eta} * u_E - \frac{\partial x}{\partial \eta} * v_E\right)]$$

Όπου $v_p = \frac{v_W + v_{WW} + v_W + v_P}{4}$, $v_e = \frac{v_N + v_{NE} + v_E + v_p}{4}$

$$D_e = \frac{\mu}{J} * q_1 + \frac{\mu}{J} * q_2 \frac{\partial u}{\partial \eta} | w = \frac{\mu}{J} * [q_1 + q_2(\frac{\frac{u_N + u_{NE} + u_P + u_E}{4} \_ \frac{u_E + u_P + u_S + u_{SE}}{4}}{2})$$

$$F_n = \frac{1}{2}\rho[\left(\frac{\partial x}{\partial \xi} * v_N - \frac{\partial y}{\partial \xi} * u_N\right) - \left(\frac{\partial x}{\partial \xi} * v_p - \frac{\partial u}{\partial \xi} * u_p\right)]$$

Όπου $v_n = \frac{v_{NP} + v_{NW} + v_{WW} + v_N}{4}$, $v_p = \frac{v_{WW} + v_P + v_N + v_W}{4}$

$$D_n = \frac{\mu}{J} * q_3 * \frac{\partial u}{\partial \eta} - \frac{\mu}{J} * q_2 = \frac{\mu}{J} * [q_3 + q_2(\frac{\frac{u_N + u_{NW} + u_P + u_W}{4} \_ \frac{u_N + u_{WE} + u_P + u_E}{4}}{2 * \Delta\xi})$$

$$F_s = \frac{1}{2}\rho[\left(\frac{\partial x}{\partial \xi} * v_P - \frac{\partial y}{\partial \xi} * u_p\right) - \left(\frac{\partial x}{\partial \xi} * v_s - \frac{\partial u}{\partial \xi} * u_s\right)]$$

Όπου $v_N = \frac{v_{NW} + v_N + v_P + v_W}{4}$, $v_P = \frac{v_W + v_P + v_{SW} + v_S}{4}$

$$D_s = \frac{\mu}{J} * q_3 | s - \frac{\mu}{J} * q_2 \frac{\partial u}{\partial \xi} | s = \frac{\mu}{J} * [q_3 - q_2(\frac{\frac{u_S + u_{SW} + u_P + u_W}{4} \_ \frac{u_S + u_{SE} + u_P + u_E}{4}}{2 * \Delta\xi})$$

Παρατηρούμε, ότι για τις τιμές των εδρών, χρησιμοποιείται ένας μέσος όρος των περιφερειακών τιμών των ταχυτήτων που μελετάμε. Και ανάλογα με την θέση της έδρας και την ζητούμενη ταχύτητα σε αυτή (u, v), παίρνουμε έναν μέσο όρο 2 ή 4 σημείων. Σε ένα πολύ μεταγενέστερο βήμα, θα μπορούσε κανείς να πάρει έναν σταθμισμένο μέσο όρο, ανάλογα με το σημείο που έχει την μεγαλύτερη επίδραση στην υπό μελέτη έδρα, αλλά στα πλαίσια της εργασίας αυτής, επιλέχθηκε ένας απλός μέσος όρος.

Ενώ αντίστοιχα στην v διεύθυνση

$$a_P v_P = a_E v_E + a_W v_W + a_N v_N + a_S v_S + S_v$$

Όπου

$$a_e = -F|_e + D|_e$$
$$a_w = \mathrm{F}|_w + \mathrm{D}|_w$$
$$a_n = -F|_n + D\mathrm{I}_n$$
$$a_s = \mathrm{F}|_s + \mathrm{D}|_s$$

$$a_P = a_e + a_w + a_n + a_s + S_p$$

Και οι αντίστοιχοι συντελεστές α,

$$F_w = \frac{1}{2}\rho[\left(\frac{\partial y}{\partial \eta} * u_W - \frac{\partial x}{\partial \eta} * v_W\right) - \left(\frac{\partial y}{\partial \eta} * u_P - \frac{\partial x}{\partial \eta} * v_P\right)]$$

Όπου $u_W = \frac{u_W + u_{SW} + u_P + u_S}{4}$, $u_P = \frac{u_P + u_E + u_S + u_E}{4}$

$$D_w = \frac{\mu}{J}[q_1 \left| w + q_2 \frac{\partial v}{\partial \eta} \right| w] = \frac{\mu}{J} * [q_2 \frac{(v_w + v_{SW} + v_S + v_P) - (v_{NW} + v_N + v_P + v_w)}{8}]$$

$$F_E = \frac{1}{2}\rho[\left(\frac{\partial y}{\partial \eta} * u_E - \frac{\partial y}{\partial \eta} * v_E\right) - \left(\frac{\partial y}{\partial \eta} * u_P - \frac{\partial x}{\partial \eta} * v_P\right)]$$

Όπου $u_E = \frac{u_W + u_{EE} + u_{SEE} + u_{SE}}{4}$, $u_P = \frac{u_P + u_E + u_S + u_{SE}}{4}$

$$D_e = \frac{\mu}{J}[q_1 \left|e - q_2 \frac{\partial v}{\partial \eta}\right| e = \frac{\mu}{J}[-q_3 + q_2 \frac{(v_{NW} + v_W + v_W + v_P) + (v_N + v_P + v_E + v_{NE})}{8}]$$

$$F_S = \frac{1}{2}\rho[\left(\frac{\partial x}{\partial \xi} * v_P - \frac{\partial y}{\partial \xi} * u_P\right) - \left(\frac{\partial x}{\partial \xi} * v_S - \frac{\partial u}{\partial \xi} * u_S\right)]$$

Όπου $u_P = \frac{u_P + u_E + u_S + u_{SE}}{4}$, $u_S = \frac{u_{SE} + u_S + u_{SS} + u_{SSE}}{4}$

$$D_s = \frac{\mu}{J}[q_3 + q_2 \frac{\partial v}{\partial \xi}|s = \frac{\mu}{J} * [q_3 + q_2 \frac{(v_w + v_{SW} + v_S + v_P) - (v_{SE} + v_S + v_P + v_E)}{8}]$$

$$F_N = -\frac{1}{2}\rho[\left(\frac{\partial x}{\partial \xi} * v_P - \frac{\partial y}{\partial \xi} * u_p\right) - \left(\frac{\partial x}{\partial \xi} * v_s - \frac{\partial u}{\partial \xi} * u_s\right)]$$

Όπου $u_P = \frac{u_P + u_E + u_S + u_{SE}}{4}$, $u_S = \frac{u_N + u_{NE} + u_E + u_E}{4}$

$$D_n = \frac{\mu}{J}[q_3 \left|n + q_2 \frac{\partial v}{\partial \xi}\right| n = \frac{\mu}{J} * [-q_3 + q_2 \frac{(v_{NW} + v_W + v_N + v_P) + (v_N + v_E + v_P + v_{NE})}{8}]$$

Τέλος, οι εξισώσεις του όρου πηγής (της μεταβολής της πίεσης) είναι

$$S_u = \frac{\partial y}{\partial \eta} p|_w - \frac{\partial y}{\partial \eta} p|_e + \frac{\partial y}{\partial \xi} p|_n - \frac{\partial y}{\partial \xi} p|_s$$

Και

$$S_v - \frac{\partial x}{\partial \eta} p|_w + \frac{\partial x}{\partial \eta} p|_e - \frac{\partial x}{\partial \xi} p|_n + \frac{\partial x}{\partial \xi} p|_s$$

Τέλος, ο όρος πηγής της διατμητικής τάσης στην y κατεύθυνση είναι:

$$S_P = -\frac{\mu}{\Delta y_P} A_{cell}$$

Επίλυση

Για την επίλυση των εξισώσεων των ορμών, θα χρησιμοποιήσουμε τον αλγόριθμο Thomas, για τριδιαγώνια συστήματα. Συγκεκριμένα, θα λύσουμε μια φορά σκανάροντας το πλέγμα μας κατά την x κατεύθυνση, σειρά ανά σειρά, και μια φορά κατά την y κατεύθυνση σκανάροντας το πλέγμα ανά στήλη.

Οι πίνακες θα γεμίζουν ως εξής:

Για την u ορμή

$$-a_w u_w + a_p u_p - a_e u_e = a_{n^u n} + a_s u_s + S_u$$

Για την v ορμή

$$-a_{n^u n} + a_p u_p - a_s u_s = a_e u_e + a_w u_w + S_v$$

Σε κάθε βήμα μέσα στην επανάληψη, θεωρούμε γνωστές τις τιμές των α_s και α_w από την προηγούμενη επανάληψη και την προηγούμενη σειρά αντίστοιχα.

Τελικά, έχουμε υπολογίσει το μη διορθωμένο πεδίο ταχυτήτων, το οποίο τώρα καλούμαστε να διορθώσουμε μέσω του αλγορίθμου SIMPLE

Ο αλγόριθμος SIMPLE

Έχοντας υπολογίσει πλέον το πεδίο ταχυτήτων, επανερχόμαστε στους κεντρικούς p όγκους ελέγχου, όπου καλούμαστε να υπολογίσουμε τις τιμές της διορθωμένης πίεσης p'

Γνωρίζουμε ότι το πεδίο ταχυτήτων που υπολογίσαμε, έχοντας άγνωστο το πεδίο της πίεσης εξαρχής, δεν μπορεί να ικανοποιεί την εξίσωση της συνέχειας. Οπότε, για κάθε σημείο p, των όγκων ελέγχου, γνωρίζουμε ότι η τιμή που υπολογίσαμε, από την τιμή που ικανοποιεί την συνέχεια, θα απέχει μια τιμή u και v.

$$\boxed{u = u^* + u'}$$
$$\boxed{v = v^* + v'}$$
$$\boxed{p = p^* + p'}$$

Αντικαθιστώντας τις εξισώσεις αυτές στις εξισώσεις τις ορμής, και η έπειτα αντικατάσταση τους στην εξίσωση της συνέχειας, μας δίνει μια εξίσωση της μορφής:

$$a_P p'_P = a_E p'_E + a_W p'_W + a_N p'_N + a_S p'_S + S_{p'}$$

Με

$$a_E = \varrho\left(\frac{y_\eta^2}{a_{u_p}} + \frac{x_\eta^2}{a_{v_p}}\right)_e$$
$$a_W = \rho\left(\frac{y_\eta^2}{a_{u_p}} + \frac{x_\eta^2}{a_{v_p}}\right)_w$$
$$a_N = \rho\left(\frac{y_\xi^2}{a_{u_p}} + \frac{x_\xi^2}{a_{v_p}}\right)_n$$
$$a_s = e\left(\frac{y_\xi^2}{a_{u_p}} + \frac{x_\xi^2}{a_{v_s}}\right)_s$$
$$a_p = a_E + a_W + a_N + a_s$$
$$S_{p'} = \varrho\left(\left(u\frac{\partial y}{\partial \eta} - v\frac{\partial x}{\partial \eta}\right)^*_w - \left(u\frac{\partial y}{\partial \eta} - v\frac{\partial x}{\partial \eta}\right)^*_e + \left(v\frac{\partial x}{\partial \xi} - u\frac{\partial y}{\partial \xi}\right)^*_s - \left(v\frac{\partial x}{\partial \xi} - u\frac{\partial y}{\partial \xi}\right)^*_n\right)$$

Και θα διακριτοποιήσουμε τις τιμές και πάλι αντίστοιχα με διακριτοποιημένες τιμές των εξισώσεων της ορμής.

Παρατηρούμε ότι αυτήν την φορά, ως όρο πηγής, έχουμε την ασυνέχεια που έχει δημιουργηθεί λόγω των διαφορών των πραγματικών τιμών των ταχυτήτων, από αυτές που υπολογίσαμε αρχικά, και αυτό θα είναι τελικά και το κριτήριο σύγκλισής του προγράμματος ολόκληρου. Όταν δηλαδή θα έχουμε μια τιμή μικρότερη από 10^{-4} στην διόρθωση (και άρα στην πλήρωση της συνέχειας) τότε η επίλυσή μας θα τερματίζει.

Έχοντας υπολογίσει το διορθωμένο πλέον πεδίο πίεσης, πρέπει να υπολογισθεί και το νέο πεδίο ταχύτητας με βάση τις πιέσεις αυτές.

$$p_{l,J} = p^*_{l,j} + p'_{l,J}$$
$$u_{i,J} = u^*_{i,j} + d_{i,J}(p'_{l-1,J} - p'_{l,j})$$
$$v_{l,j} = v^*_{l,j} + d_{l,j}(p'_{l,J-1} - p'_{l,j})$$

Όπου

$$d_{i+1,J} = \frac{A_{i+1,J}}{a_{i+1,H}}$$
$$d_{I,j+1} = \frac{A_{I,j+1}}{a_{I,j+1}}$$

Κατά τις διορθώσεις των βαθμωτών μεγεθών έγινε εισαγωγή και μια τιμή υποχαλάρωσης ω, για να μπορέσουμε να ελέγξουμε καλύτερα την σύγκλιση του αλγορίθμου και να αποφύγουμε μεγάλες διαταραχές στην εξέλιξη του κώδικα.

Οριακές συνθήκες

Οριακές συνθήκες εξισώσεων ταχύτητας

Οι οριακές συνθήκες της ταχύτητας χωρίζονται σε 3 μέρη:

Αυτές που είναι σχετικές με την είσοδο, αυτές που έχουν σχέση με την έξοδο και αυτές που έχουν σχέση με τα τοιχώματα.

Ορ. Συνθήκες εισόδου

Στο συγκεκριμένο πρόβλημα έχουμε ως είσοδο ένα διάνυσμα ταχύτητας κάθετο στη επιφάνεια του όγκου ελέγχου εισόδου. Ως γενικό κανόνα, δεν επιλύουμε στον πρώτο όγκο ελέγχου και άρα και πρώτο σημείο του τοιχώματος. Αντ' αυτού, αποθηκεύουμε σε αυτό τις τιμές u και v εισόδου, και ακριβώς κατάντη αυτού, ξεκινάει η επίλυσή μας. Δεν χρειάζεται κάποια περεταίρω τροποποίηση ο κώδικάς μας, αφού παίρνει απευθείας τιμή από το κελί αυτό.

Οριακές Συνθήκες εξόδου

Όπως και στην είσοδο, στην έξοδο επίσης δεν πραγματοποιούμε επίλυση. Αντ' αυτού, επιλύουμε τις τιμές των u και v ταχυτήτων μέχρι την n-1 τιμή του πλέγματος, έχοντας προσέξει βέβαια, η έξοδος να είναι αρκετά μακριά από το όποιο πεδίο διαταραχών ώστε η ροή να έχει αναπτυχθεί πλήρως και να μην έχουμε αναμενόμενες μεταβολές στις τιμές των βαθμωτών μεγεθών μας. Η επόμενη κίνησή μας, είναι να αποθηκεύσουμε στην τελευταία θέση του πλέγματός μας, ίδια τιμή με την προ τελευταία.

Τέλος, για να σιγουρέψουμε ότι η ολική συνέχεια του συστήματός μας διατηρείται, είναι ανάγκη να πολλαπλασιάσουμε τον όρο αυτόν της u ή v ταχύτητας εξόδου, με τον λόγο συνολικής ροής μάζας εισόδου-εξόδου διορθώνοντας με έναν σχετικά παράδοξο τρόπο την ασυνέχεια που ίσως τυχών να έχει δημιουργηθεί κατά την πορεία του αλγορίθμου.

Οριακές. συνθήκες τοιχώματος

Στο τοίχωμα, πρέπει να προσέξουμε ότι αρχικά οι ταχύτητες u και v είναι 0 και αυτό δεν πρέπει να αλλάζει κατά την πορεία του αλγορίθμου. Επίσης, έχουμε πλέον και έναν έξτρα όρο πηγής, αυτόν της διατμητικής τάσης που αναπτύσσεται στο τοίχωμα:

Ο όρος πηγής αυτός όμως, αποθηκεύεται μέσα στον όρο ap, διότι επίσης εξαρτάται από την προς επίλυση ταχύτητα.

Οριακές συνθήκες εξισώσεων πίεσης

Στην επίλυση της πίεσης, για κάθε όρο που δεν χρειάζεται διόρθωση (είσοδος, έξοδος, τοίχωμα) αποκόπτουμε την επαφή της εξίσωσης με τον όρο αυτό α=0

Συνεπώς, για την είσοδο, έχουμε αw=0, για την έξοδο έχουμε αe=0 ενώ για το τοίχωμα έχουμε αn ή αs=0, ανάλογα αν είμαστε στο πάνω ή στο κάτω μέρος του τοιχώματος.

Επίσης, στον όρο της πίεσης, η παράγωγος dp/dn κοντά στο τοίχωμα είναι επίσης ίση με το 0.

Τέλος, κατά την διάρκεια της διόρθωσης της πίεσης, όσα μεγέθη μας είναι ήδη γνωστά (ταχύτητες εισόδου, εξόδου, τοιχωμάτων) συνεχίζουν να παραμένουν όσο οι αρχικές τιμές και δεν χρειάζονται διόρθωση.

Ρεϊζόπουλος Μάριος ΑΕΜ: 5782
Τερζής Δημήτριος ΑΕΜ:6101

Σχόλια:

Για την περάτωση της εργασίας δαπανήθηκε αρκετό μέρος του χρόνου τόσο στην μετατροπή όσο και στην διακριτοποίηση των εξισώσεων. Επίσης, αντιμετωπίστηκαν δυσκολίες κατά την αντιστοίχιση των πλεγμάτων και των σημείων επειδή τα σημεία p-Control Volume είναι λιγότερα από τα v, u. Παρά τις δυσκολίες ο βασικός κορμός την εργασίας έχει δημιουργηθεί. Επίσης έχουν περαστεί και οι οριακές συνθήκες. Μένει να γίνει αντιστοίχιση των μεγεθών και η διόρθωση της πίεσης (Pressure Correction). Όσον αφορά τους πίνακες των αποτελεσμάτων, δείχνουν μια καλή τάση στην ροή κοντά στο τοίχωμα όπου θέλουμε να πιάσουμε την εξέλιξη του οριακού στρώματος και επιπλέον κατά την εναλλαγή u-v καθώς στο οριζόντιο τμήμα του αγωγού είναι σημαντικότερη η u και στο κάθετο τμήμα του αγωγού είναι σημαντικότερη η v. Φαίνεται ότι το παραπάνω είναι κοντά στην επίτευξη του:

Βιβλιογραφία

1) Wei Shyy, Madhukar Rao, "Computational Fluid Dynamics with moving Boundaries"
2) Ferziger, Peric "Computational Methods for Fluid Dynamics"
3) H.K. Versteeg, W. Malalasekera "An Introduction to Computational Fluid Dynamics"
4) John D. Anderson Jr., “Mc-Graw Hills series Computational fluid mechanics”
5) Σημειώσεις μαθήματος “Υπολογιστική Ρευστομηχανική”

Code Listing

Αρχείο με όνομα metrics

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from pylab import meshgrid

# Mesh reading
mesh=np.loadtxt('mesh.dat')
x=np.zeros(0)
y=np.zeros(0)
x=np.append(x,mesh[:,0])
y=np.append(y,mesh[:,1])

#Array initialization

xx=np.zeros((200, 60))
yy=np.zeros((200, 60))
xp=np.zeros((200, 60))
yp=np.zeros((200, 60))
xu=np.zeros((200, 60))
yu=np.zeros((200, 60))
xv=np.zeros((200, 60))
yv=np.zeros((200, 60))
x_y=np.zeros((60, 200))
y_y=np.zeros((60, 200))


b11=np.zeros((200, 60))
b12=np.zeros((200, 60))
b21=np.zeros((200, 60))
b22=np.zeros((200, 60))
b11w=np.zeros((200, 60))
b12w=np.zeros((200, 60))
b21w=np.zeros((200, 60))
b22w=np.zeros((200, 60))
```

```
b11s=np.zeros((200, 60))
b12s=np.zeros((200, 60))
b21s=np.zeros((200, 60))
b22s=np.zeros((200, 60))
Jp=np.zeros((200, 60))
Ju=np.zeros((200, 60))
Jv=np.zeros((200, 60))
volp=np.zeros((200, 60))
volu=np.zeros((200, 60))
volv=np.zeros((200, 60))
pcheck=np.zeros((200, 60))
ucheck=np.zeros((200, 60))
vcheck=np.zeros((200, 60))

#second order central
b11ksi=np.zeros((200, 60))
b11hta=np.zeros((200, 60))
b12ksi=np.zeros((200, 60))
b12hta=np.zeros((200, 60))
b21ksi=np.zeros((200, 60))
b21hta=np.zeros((200, 60))
b22ksi=np.zeros((200, 60))
b22hta=np.zeros((200, 60))

#second order west
b11ksiw=np.zeros((200, 60))
b11htaw=np.zeros((200, 60))
b12ksiw=np.zeros((200, 60))
b12htaw=np.zeros((200, 60))
b21ksiw=np.zeros((200, 60))
b21htaw=np.zeros((200, 60))
b22ksiw=np.zeros((200, 60))
b22htaw=np.zeros((200, 60))

#Second order south
b11ksis=np.zeros((200, 60))
b11htas=np.zeros((200, 60))
b12ksis=np.zeros((200, 60))
b12htas=np.zeros((200, 60))
b21ksis=np.zeros((200, 60))
b21htas=np.zeros((200, 60))
b22ksis=np.zeros((200, 60))
b22htas=np.zeros((200, 60))



# Grid points
a=0
while a <len(x):
    for j in np.arange(0, 60):
        for i in np.arange(0, 200):
            xx[i, j] = x[a]
            yy[i, j] = y[a]
            a += 1

# Reading by y
b=0
while b <len(y):
    for i in np.arange(0, 60):
```

```
        for j in np.arange(0, 200):
            x_y[i, j] = x[b]
            y_y[i, j] = y[b]
            b += 1


# Center of cell - Pressure Points
for i in range(1, len(xx[:, 0])):
    for j in range(1, len(xx[0, :])):
        xp[i-1, j-1] = 0.25*(xx[i-1, j-1]+xx[i, j-1]+xx[i, j]+xx[i-1, j])
        yp[i-1, j-1] = 0.25*(yy[i-1, j-1]+yy[i, j-1]+yy[i, j]+yy[i-1, j])
# Correction for the last row
xp[-1, :] = xp[-2, :]
yp[-1, :] = yp[-2, :]

# u-cell ( West face)

for i in range(len(xx[:, 0])):
    for j in range(1, len(xx[0, :])):
        xu[i, j-1] = 0.5*(xx[i, j-1]+xx[i, j])
        yu[i, j-1] = 0.5*(yy[i, j-1]+yy[i, j])

# v-cell ( South face)

for i in range(1, len(xx[:, 0])):
    for j in range(len(xx[0, :])):
        xv[i-1, j]=0.5*(xx[i-1, j]+xx[i, j])
        yv[i-1, j]=0.5*(yy[i-1, j]+yy[i, j])

# Correction for the last row
xv[-1, :] = xv[-2, :]
yv[-1, :] = yv[-2, :]

plt.figure()
plt.title("όγκοι ελέγχου physical plane")
# for i in range(0, len(xp[:, 0])):
#     plt.plot(xv[i, :-1], yv[i, :-1], c='red')
# plt.plot(xv, yv, c='red')

for i in range(0, len(xv[:, 0])):
    plt.plot(xu[i, :-1], yu[i, :-1], c='green')
plt.plot(xu, yu, c='green')
plt.scatter(xp, yp, c='black', s=1)

# Staggered grids
# plt.plot(xx, yy, c='black')
# plt.plot(x_y, y_y, c='black')
# plt.figure()
for i in range(0, len(xv[:, 0])):
    plt.plot(xu[i, :-1], yu[i, :-1], c='black')
plt.plot(xv, yv, c='black')
plt.xlabel('x')
plt.ylabel('y')

plt.show()
# Metrics and Jacobians

# Center of c.v.
for i in np.arange(1,199):
```

```
    for j in np.arange(1,59):
        b11[i, j] = xu[i+1,j] - xu[i, j]
        b22[i, j] = yv[i, j+1]-yv[i, j]
        b12[i, j] = yu[i+1, j]-yu[i, j]
        b21[i, j] = xv[i, j+1]-xv[i, j]

        # West face of c.v.
        b11w[i, j] = xp[i, j]-xp[i-1, j]
        b22w[i, j] = yy[i, j+1]-yy[i, j]
        b12w[i, j] = yp[i, j]-yp[i-1, j]
        b21w[i, j] = xx[i, j+1]-xx[i, j]
        if i==1:
            b11w[i, j] = -3*xu[i,j]+4*xp[i,j]-xu[i+1,j]
            b12w[i, j] = -3*yu[i,j]+4*yp[i,j]-yu[i+1,j]

        if i==-1:
            b11w[i+1,j] = 3*xu[i,j]-4*xp[i-1,j]+xu[i-1,j]
            b12w[i+1,j] = 3*yu[i,j]-4*yp[i-1,j]+yu[i-1,j]
            b22w[i+1,j] = yy[i+1,j+1]-yy[i+1,j]
            b21w[i+1,j] = xx[i+1,j+1]-xx[i+1,j]

        # South face of c.v
        b11s[i,j] = xx[i+1,j]-xx[i,j]
        b22s[i,j] = yp[i,j]-yp[i,j-1]
        b12s[i,j] = yy[i+1,j]-yy[i,j]
        b21s[i,j] = xp[i,j]-xp[i,j-1]
        if i==1:
            b22s[i,j] = -3*yv[i,j]+4*yp[i,j]-yv[i,j+1]
            b21s[i,j] = -3*xv[i,j]+4*xp[i,j]-xv[i,j+1]
        if i==-1:
            b11s[i,j+1] = xx[i+1,j+1]-xx[i,j+1]
            b12s[i,j+1] = yy[i+1,j+1]-yy[i,j+1]
            b21s[i,j+1] = 3*xv[i,j]-4*xp[i,j-1]+xv[i,j-1]
            b22s[i,j+1] = 3*yv[i,j]-4*yp[i,j-1]+yv[i,j-1]

        # Special treatment on Boundaries / south
        if i==199:
            b11s[i+1,j] = 0
            b12s[i+1,j] = 0
            b21s[i+1,j] = xu[i+1,j]-xu[i,j]
            b22[i+1,j] = yu[i+1,j]-yu[i,j]

        # Second order metrics
        #Pressure points

        # d2x/dksi2
        b11ksi[i,j] = b11[i+1,j]-b11[i,j]
        #d2x/dksidhta
        b11hta[i,j] = b11[i,j+1]-b11[i,j]
        #d2y/dksi2
        b12ksi[i,j] = b12[i+1,j]-b12[i,j]
        #d2y/dksidhta
        b12hta[i,j] = b12[i,j+1]-b12[i,j]
        #d2x/dksidhta
        b21ksi[i,j] = b21[i+1,j]-b21[i,j]
        #d2x/dhta2
        b21hta[i,j] = b21[i,j+1]-b21[i,j]
        #d2y/dksidhta
        b22ksi[i,j] = b22[i+1,j]-b22[i,j]
```

```
        #d2y/dhta2
        b22hta[i,j] = b22[i,j+1]-b22[i,j]


        # u face
        # d2x/dksi2
        b11ksiw[i, j] = b11w[i + 1, j] - b11w[i, j]
        # d2x/dksidhta
        b11htaw[i, j] = b11w[i, j + 1] - b11w[i, j]
        # d2y/dksi2
        b12ksiw[i, j] = b12w[i + 1, j] - b12w[i, j]
        # d2y/dksidhta
        b12htaw[i, j] = b12w[i, j + 1] - b12w[i, j]
        # d2x/dksidhta
        b21ksiw[i, j] = b21w[i + 1, j] - b21w[i, j]
        # d2x/dhta2
        b21htaw[i, j] = b21w[i, j + 1] - b21w[i, j]
        # d2y/dksidhta
        b22ksiw[i, j] = b22w[i + 1, j] - b22w[i, j]
        # d2y/dhta2
        b22htaw[i, j] = b22w[i, j + 1] - b22w[i, j]


        #v-face
        # d2x/dksi2
        b11ksis[i, j] = b11s[i + 1, j] - b11s[i, j]
        # d2x/dksidhta
        b11htas[i, j] = b11s[i, j + 1] - b11s[i, j]
        # d2y/dksi2
        b12ksis[i, j] = b12s[i + 1, j] - b12s[i, j]
        # d2y/dksidhta
        b12htas[i, j] = b12s[i, j + 1] - b12s[i, j]
        # d2x/dksidhta
        b21ksis[i, j] = b21s[i + 1, j] - b21s[i, j]
        # d2x/dhta2
        b21htas[i, j] = b21s[i, j + 1] - b21s[i, j]
        # d2y/dksidhta
        b22ksis[i, j] = b22s[i + 1, j] - b22s[i, j]
        # d2y/dhta2
        b22htas[i, j] = b22s[i, j + 1] - b22s[i, j]

        # Jacobians

        Jp[i,j]=b11[i,j]*b22[i,j]-b12[i,j]*b21[i,j]
        Ju[i, j] = b11w[i, j] * b22w[i, j] - b12w[i, j] * b21w[i, j]
        Jv[i, j] = b11s[i, j] * b22s[i, j] - b12s[i, j] * b21s[i, j]

        #Volumes of c.v

        #Pressure points
        volp[i,j] = abs(0.5*((xx[i+1,j+1]-xx[i,j])*(yy[i,j+1]-yy[i+1,j])-(xx[i,j+1]-
xx[i+1,j])*(yy[i+1,j+1]-yy[i,j])))

        #u points
        volu[i,j] = abs(0.5*((xv[i+1,j+1]-xv[i,j])*(yv[i,j+1]-yv[i+1,j])-(xv[i,j+1]-
xv[i+1,j])*(yv[i+1,j+1]-yv[i,j])))

        #v points
        volv[i,j] = abs(0.5*((xu[i+1,j+1]-xu[i,j])*(yu[i,j+1]-yu[i+1,j])-(xu[i,j+1]-
```

```
xu[i+1,j])*(yu[i+1,j+1]-yu[i,j])))


for i in np.arange(1, 199):
    for j in np.arange(1, 59):
        pcheck[i,j] = Jp[i,j]-volp[i,j]
        ucheck[i,j] = Ju[i,j]-volu[i,j]
        vcheck[i,j] = Jv[i,j]-volv[i,j]




# Dataframes

# df1=pd.DataFrame(Jp)
# df1.to_excel('Jp.xlsx')
# df2=pd.DataFrame(Ju)
# df2.to_excel('Ju.xlsx')
# df3=pd.DataFrame(Jv)
# df3.to_excel('Jv.xlsx')
# df4=pd.DataFrame(pcheck)
# df4.to_excel('pcheck.xlsx')
# df5=pd.DataFrame(ucheck)
# df5.to_excel('ucheck.xlsx')
# df6=pd.DataFrame(vcheck)
# df6.to_excel('vcheck.xlsx')
```

Αλγόριθμος με όνομα solver

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import time

from metrics import *

dens = 1.23
visc = 18.37e-6
omega = 0.5

Nx = 200
Ny = 60

u_old = np.zeros([len(xu[:, 0]), len(yu[0, :])])
u_new = np.zeros([len(xu[:, 0]), len(yu[0, :])])

v_old = np.zeros([len(xv[:, 0]), len(yv[0, :])])
v_new = np.zeros([len(xv[:, 0]), len(yv[0, :])])

v_u_p = np.zeros([len(xu[:, 0]), len(yu[0, :])])
u_v_p = np.zeros([len(xv[:, 0]), len(yv[0, :])])

p_old = np.zeros([len(xp[:, 0]), len(yp[0, :])])
p_new = np.zeros([len(xp[:, 0]), len(yp[0, :])])


# arxikes sun8ikes
```

```
for i in range(0, len(xu[:, 0])):
    if yy[i, -1] - yy[0, -1] < 0.00000001:
        u_old[i, 1:-1] = 0.1
        u_new[i, 1:-1] = 0.1
        v_old[i, 1:-1] = 0
        v_new[i, 1:-1] = 0
    elif np.abs(xx[i, 0] - xx[i-1, 0]) < 0.0000001:
        u_old[i, 1:-1] = 0
        u_new[i, 1:-1] = 0
        v_old[i, 1:-1] = 0.1
        v_new[i, 1:-1] = 0.1
    else:
        u_old[i, 1:-1] = np.sqrt(2)/2 * 0.1
        u_new[i, 1:-1] = np.sqrt(2)/2 * 0.1
        v_old[i, 1:-1] = np.sqrt(2)/2 * 0.1
        v_new[i, 1:-1] = np.sqrt(2)/2 * 0.1



q1 = b21hta + b22hta
q2 = b11hta + b12hta
q3 = b11ksi + b12ksi

q1w = b21htaw + b22htaw
q2w = b11htaw + b12htaw
q3w = b11ksiw + b12ksiw

q1s = b21htas + b22htas
q2s = b11htas + b12htas
q3s = b11ksis + b12ksis

De_u = []
Fe_u = []
ae_u = []
Dw_u = []
Fw_u = []
aw_u = []
Dn_u = []
Fn_u = []
an_u = []
Ds_u = []
Fs_u = []
as_u = []
dp_u_x = []
dp_u_y = []

Sp_u = []

ap_u_big = np.zeros([0, Ny-2])
ap_v_big = np.zeros([0, Ny-2])

for i in range(1, len(xu[:, 0]) - 2):
    ap_u = []
    ae_u = []
    aw_u = []
    an_u = []
    as_u = []
    dp_u = []
    Sp_u = []
```

```
    for j in range(1, len(yu[0, :]) - 1):
        # u
        v_u_p[i, j] = (v_old[i, j+1] + v_new[i-1, j+1] + v_new[i-1, j] + v_old[i, j]) / 4

        # west
        Dw_u = visc / Ju[i, j] * (q1w[i, j] +
                                  q2w[i, j] * ((u_new[i-1, j+1] + u_old[i, j+1] + u_old[i,
j] + u_new[i-1, j]) / 4 -
                                               (u_new[i-1, j] + u_old[i, j] + u_new[i-1, j-
1] + u_new[i, j-1]) / 4) / 2)
        Fw_u = 0.5 * dens * (
                    b22w[i, j] * (u_old[i, j] - u_new[i-1, j]) - b21w[i, j] * (v_u_p[i, j] +
v_u_p[i+1, j]))
        aw_u = np.append(aw_u, Fw_u + Dw_u)

        # east
        De_u = visc / Ju[i, j] * (q1w[i, j] -
                              q2w[i, j] * ((u_old[i, j+1] + u_old[i+1, j+1] + u_old[i+1, j] +
u_old[i, j]) / 4 -
                                           (u_old[i+1, j] + u_old[i, j] + u_new[i, j-1]
+ u_old[i+1, j-1]) / 4) / 2)
        Fe_u = - 0.5 * dens * (b22w[i, j] * (- u_old[i, j] + u_old[i+1, j]) - b21w[i, j] *
(-v_u_p[i, j] + v_u_p[i+1, j]))
        ae_u = np.append(ae_u, Fe_u + De_u)


        # north
        Dn_u = visc / Ju[i, j] * (q3w[i, j]-
                                  q2w[i, j] * (-(u_new[i-1, j+1] + u_old[i, j] +
u_old[i, j+1] + u_new[i-1, j]) / 4 +
                                               (u_old[i, j] + u_old[i, j+1] + u_old[i+1, j+1] +
u_old[i+1, j]) / 4) / 2)
        Fn_u = - 0.5 * dens * (b11w[i, j] * (- v_u_p[i, j] + v_u_p[i, j+1]) - b12w[i, j] *
(- u_old[i, j] + u_old[i, j+1]))
        an_u = np.append(an_u, Fn_u + Dn_u)

        # south
        Ds_u = visc / Ju[i, j] * (q3w[i, j] +
                                  q2w[i, j] * ((u_new[i-1, j] + u_old[i, j] + u_new[i-1,
j-1] + u_new[i, j-1]) / 4 -
                                               (u_old[i, j] + u_old[i+1, j] + u_old[i+1, j-1] +
u_new[i, j-1]) / 4) / 2)
        Fs_u = 0.5 * dens * (b11w[i, j] * (v_u_p[i, j] - v_u_p[i, j-1]) - b12w[i, j] *
(u_old[i, j] - u_new[i, j-1]))
        as_u = np.append(as_u, Fs_u + Ds_u)

        dp_u = np.append(dp_u_x, b22w[i, j] * p_old[i+1, j] - b22w[i, j] * p_old[i-1, j] +
                         b12w[i, j] * p_old[i, j+1] - b12w[i, j] * p_old[i, j-1])
        if j == 1 or j == Ny - 1:
            Sp_u = np.append(Sp_u, - b12w[i, j] * visc)
        else:
            Sp_u = np.append(Sp_u, 0)

    ap_u = aw_u + ae_u + an_u + as_u
    ap_u_big = np.vstack([ap_u_big, ap_u])
    # Thomas for
    A = -an_u
    B = ap_u
    C = -as_u
```

```
    K = dp_u + ae_u + aw_u + Sp_u
    # K[-1] = K[-1] - C[-1] * u[-1] kapoiou eidous or. sun8iki
    #Thomas
    for k in range(1, len(A)):
        mc = A[k] / B[k - 1]
        B[k] = B[k] - mc * C[k - 1]
        K[k] = K[k] - mc * K[k - 1]
    Thomas = A.copy()
    Thomas[-1] = K[-1] / B[-1]
    for l in range(len(A)-2, -1, -1):
        Thomas[l] = (K[l] - C[l] * Thomas[l + 1]) / B[l]
    for l in range(1, Ny - 2):
        u_new[i, l] = Thomas[l - 1]


aa = pd.DataFrame(u_old)
bb = pd.DataFrame(u_new)
aa.to_excel('aa.xlsx')
bb.to_excel('bb.xlsx')
print(u_new)



for i in range(1, len(xv[:, 0]) - 2):
    ap_v = []
    ae_v = []
    aw_v = []
    an_v = []
    as_v = []
    dp_v = []
    Sp_v = []
    for j in range(1, len(yv[0, :]) - 1):
        # v
        u_v_p[i, j] = (u_old[i, j + 1] + u_new[i-1, j + 1] + u_new[i-1, j] + u_old[i, j]) /
4
        # west
        Dw_v = visc / Jv[i, j] * (q1s[i, j] +
                                  q2s[i, j] * (-(v_new[i-1, j] + v_new[i-1, j-1] +
v_new[i, j-1] + v_old[i, j]) / 4 +
                                               (v_new[i-1, j+1] + v_old[i, j] + v_old[i, j+1] +
v_new[i-1, j]) / 4) / 2)
        Fw_v = 0.5 * dens * (b22s[i, j] * (u_v_p[i, j] - u_v_p[i-1, j]) - b21s[i, j] *
(v_old[i, j] - v_new[i-1, j]))
        aw_v = np.append(aw_v, Dw_v + Fw_v)

        # east
        De_v = De_u = visc / Jv[i, j] * (q1s[i, j] -
                                         q2s[i, j] * ((v_new[i, j-1] + v_old[i+1, j-1] +
v_old[i, j] + v_old[
                    i+1, j]) / 4 -
                                                      (v_old[i, j+1] + v_old[i, j] +
v_old[i+1, j+1] + v_old[
                                                          i+1, j]) / 4) / 2)
        Fe_v = - 0.5 * dens * (
                    b22s[i, j] * (-u_v_p[i, j] + u_v_p[i + 1, j]) - b21s[i, j] * (-v_old[i,
j] + v_old[i + 1, j]))
        ae_v = np.append(ae_v, De_v + Fe_v)

        # north
```

```
        Dn_v = visc / Jv[i, j] * (q3s[i, j] -
                                  q2s[i, j] * ((v_new[i-1, j+1] + v_old[i, j] + v_old[i,
j+1] + v_new[i-1, j]) / 4 -
                                               (v_old[i, j] + v_old[i, j+1] + v_old[i+1, j+1] +
v_old[i+1, j]) / 4) / 2)
        Fn_v = - 0.5 * dens * (b11s[i, j] * (-v_old[i, j] + v_old[i, j+1]) - b12s[i, j] * (-
u_v_p[i, j] + u_v_p[i, j+1]))
        an_v = np.append(an_v, Fn_v + Dn_v)

        # south
        Ds_v = visc / Jv[i, j] * (q3s[i, j]  +
                                  q2s[i, j] * (-(v_new[i-1, j] + v_old[i, j] + v_new[i-
1, j-1] + v_new[i, j-1]) / 4 +
                                               (v_old[i, j] + v_old[i+1, j] + v_old[i+1, j-1] +
v_new[i, j-1]) / 4) / 2)
        Fs_v = 0.5 * dens * (b11s[i, j] * (v_old[i, j] - v_new[i, j-1]) - b12s[i, j] *
(v_u_p[i, j] - v_u_p[i, j-1]))
        as_v = np.append(as_v, Ds_v + Fs_v)

        dp_v = np.append(dp_v, b22s[i, j] * p_old[i+1, j] - b22s[i, j] * p_old[i-1, j]
                         + b12s[i, j] * p_old[i, j+1] - b12s[i, j] * p_old[i, j-1])
        if j == 1 or j == Ny - 1:
            Sp_v = np.append(Sp_v, - b12s[i, j] * visc)
        else:
            Sp_v = np.append(Sp_v, 0)
    ap_v = aw_v + ae_v + an_v + as_v
    ap_v_big = np.vstack([ap_v_big, ap_v])

    A = -an_v
    B = ap_v
    C = -as_v
    K = dp_v + ae_v + aw_v + Sp_v
    # K[-1] = K[-1] - C[-1] * u[-1]
    #Thomas
    for k in range(1, len(A)):
        mc = A[k] / B[k - 1]
        B[k] = B[k] - mc * C[k - 1]
        K[k] = K[k] - mc * K[k - 1]
    Thomas = A.copy()
    Thomas[-1] = K[-1] / B[-1]
    for l in range(len(A)-2, -1, -1):
        Thomas[l] = (K[l] - C[l] * Thomas[l + 1]) / B[l]
    for l in range(1, Ny - 2):
        v_new[i, l] = Thomas[l - 1]


ae_p = []
aw_p = []
an_p = []
as_p = []




print((ap_u_big.shape))

# Pressure Correction
for i in range(1, len(xp[:, 0])-4):
    temp_e = []
```

```
    temp_w = []
    temp_n = []
    temp_s = []
    Source = []
    ae_p = []
    aw_p = []
    an_p = []
    as_p = []


    for j in range(1, len(yp[0, :]) - 2):
        if j == 1:
            if i == 1:
                temp_s = 0
                temp_n = b21[i, j] ** 2 / ap_v_big[i, j] + \
                         b11[i, j] ** 2 / (
                                 ap_u_big[i, j] + ap_u_big[i, j+1]) / 4
                temp_w = 0
                temp_e = b12[i, j] ** 2 / ap_u_big[i + 1, j] + \
                         b22[i, j] ** 2 / (0 + 0 + ap_v_big[i + 1, j] + ap_v_big[i, j]) / 4

            elif i == len(xp[:, 0])-4:
                temp_s = 0
                temp_n = b21[i, j] ** 2 / ap_v_big[i, j] + \
                         b11[i, j] ** 2 / (
                                 ap_u_big[i, j] + ap_u_big[i, j + 1]) / 2
                temp_w = b12[i, j] ** 2 / ap_u_big[i, j] + \
                         b22[i, j] ** 2 / (
                                 ap_v_big[i - 1, j] + ap_v_big[i, j] + 0 + 0) / 4
                temp_e = 0

            else:
                temp_s = 0
                temp_n = b21[i, j] ** 2 / ap_v_big[i, j] + \
                         b11[i, j] ** 2 / (
                                     ap_u_big[i, j+1] + ap_u_big[i + 1, j+1] + ap_u_big[i +
1, j ] + ap_u_big[i, j]) / 4
                temp_w = b12[i, j] ** 2 / ap_u_big[i - 1, j] + \
                         b22[i, j] ** 2 / (
                                     ap_v_big[i-1, j] + ap_v_big[i, j] + 0 + 0) / 4
                temp_e = b12[i, j] ** 2 / ap_u_big[i+1, j] + \
                         b22[i, j] ** 2 / (0 + 0 + ap_v_big[i + 1, j] + ap_v_big[i, j]) / 4
        elif j >= len(yp[0, :]) - 4:
            if i == 1:
                print(j)
                temp_s = b21[i, j] ** 2 / ap_v_big[i, j] + \
                         b11[i, j] ** 2 / (
                                     0 + ap_u_big[i + 1, j] + ap_u_big[i + 1, j - 1] + 0) /
4
                temp_n = 0
                temp_w = 0
                temp_e = b12[i, j] ** 2 / ap_u_big[i + 1, j] + \
                         b22[i, j] ** 2 / (0 + 0 + ap_v_big[i, j] + ap_v_big[i+1, j]) / 4

            elif i == len(yp[0, :]) - 4:
                temp_s = b21[i, j] ** 2 / ap_v_big[i, j] + \
                         b11[i, j] ** 2 / (
                                     ap_u_big[i, j] + 0 + 0 + ap_u_big[i, j - 1]) / 4
                temp_n = 0
```

```
                temp_w = b12[i, j] ** 2 / ap_u_big[i, j] + \
                         b22[i, j] ** 2 / (
                                 ap_v_big[i-1, j] + ap_v_big[i, j] + 0 + 0) / 4
                temp_e = 0

        else:
            print(j)

            temp_e = b12[i, j]**2 / ap_u_big[i+1, j] +\
                     b22[i, j]**2 / (ap_v_big[i, j+1] + ap_v_big[i+1, j+1] + ap_v_big[i+1,
j] + ap_v_big[i, j]) / 4

            temp_w = b12[i, j] ** 2 / ap_u_big[i - 1, j] + \
                     b22[i, j] ** 2 / (
                                 ap_v_big[i, j + 1] + ap_v_big[i - 1, j + 1] + ap_v_big[i -
1, j] + ap_v_big[i, j]) / 4
            temp_n = b21[i, j] ** 2 / ap_v_big[i, j + 1] + \
                     b11[i, j] ** 2 / (
                                 ap_u_big[i, j] + ap_u_big[i + 1, j] + ap_u_big[i + 1, j -
1] + ap_u_big[i, j - 1]) / 4

            temp_s = b21[i, j] ** 2 / ap_v_big[i, j - 1] + \
                     b11[i, j] ** 2 / (
                              ap_u_big[i, j] + ap_u_big[i + 1, j] + ap_u_big[i + 1, j + 1] +
ap_u_big[i, j + 1]) / 4
        Source = np.append(Source, dens * 0.5 * (b22[i, j] * (u_new[i+1, j] - u_new[i-1, j])
-
                           b21[i, j] *0.25 *  ((u_new[i, j] + u_new[i+1, j] + u_new[i+1,
j+1] + u_new[i, j+1]) -
                                               u_new[i, j] + u_new[i+1, j] + u_new[i+1, j-1]
+ u_new[i, j-1]) +
                           b11[i, j] * 0.25 * ((v_new[i, j] + v_new[i+1, j] + v_new[i+1,
j+1] + v_new[i, j+1]) -
                                               (v_new[i, j] + v_new[i-1, j] + v_new[i-1,
j+1] + v_new[i, j+1])) -
                           b12[i, j] * (v_new[i, j+1] - v_new[i, j])))

        ae_p = np.append(ae_p, temp_e)
        aw_p = np.append(aw_p, temp_w)
        an_p = np.append(an_p, temp_n)
        as_p = np.append(as_p, temp_s)

    ap_p = aw_p + ae_p + an_p + as_p
    A = -an_p
    B = ap_p
    C = -as_p
    K = Source + ae_p + aw_p
    # K[-1] = K[-1] - C[-1] * u[-1]
    # Thomas
    print(len(A))
    for k in range(1, len(A)):
        mc = A[k] / B[k - 1]
        B[k] = B[k] - mc * C[k - 1]
        K[k] = K[k] - mc * K[k - 1]
    Thomas = A.copy()
    Thomas[-1] = K[-1] / B[-1]
    for l in range(len(A) - 2, -1, -1):
        Thomas[l] = (K[l] - C[l] * Thomas[l + 1]) / B[l]
    print(len(Thomas))
```

```
    for l in range(1, len(A)-1):
        p_new[i, l] = Thomas[l - 1]


p_final = p_old + omega * p_new
cc = pd.DataFrame(v_new)
cc.to_excel('cc.xlsx')
dd = pd.DataFrame(p_new)
dd.to_excel('cc.xlsx')
ee = pd.DataFrame(p_final)
ee.to_excel('dd.xlsx')
```