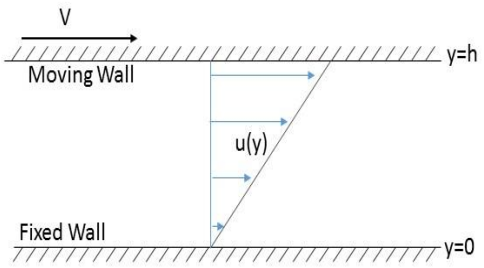
Ρεϊζόπουλος Μάριος ΑΕΜ: 5782
Τερζής Δημήτριος ΑΕΜ: 6101

2η Εργασία Υπολογιστική Ρευστομηχανική

Couette flow

Η ροή που κληθήκαμε να υπολογίσουμε και να αναλύσουμε είναι η ροή "Couette", η οποία αναφέρεται σε ροή ρευστού μεταξύ 2 παράλληλων πλακών, εκ των οποίων η μία είναι κινούμενη με μια ταχύτητα u_e. Δηλαδή εάν απόσταση μεταξύ των πλακών είναι D, τότε ισχύει $u = \begin{cases} 0, & για\ \ y = 0 \\ u, & για\ y = D \end{cases}$. Αξίζει να σημειωθεί ότι το φαινόμενο θα αναλυθεί με αρχικές συνθήκες $u = \begin{cases} 0, & για\ 0 \le y < D \\ u, & για \quad y = D \end{cases}$ και από την μόνιμη κατάσταση του φαινομένου που δείχνετε στην δίπλα εικόνα.



Η εξίσωση της ροής είναι: $\rho \frac{\partial u}{\partial t} = \mu \frac{\partial^2 u}{\partial y^2}$, και μετά την αδιαστατοποίηση γίνεται: $\frac{\partial u}{\partial t} = \frac{1}{Re_d} \frac{\partial^2 u}{\partial y^2}$. **[1]**

Επίσης, οι υπολογισμοί θα γίνουν για 3 αριθμούς Reynolds: 10, 100, 1000, και για τρεις κατανομές των σημείων *y*, με 40, 80 και 160 υπολογιστικά σημεία.

Επίσης, αξίζει να σημειωθεί ότι το timestep που θα χρησιμοποιήσουμε θα κινηθεί γύρω από τις τιμές του

$$\frac{1}{2} Re_d \cdot \Delta y^2 \textbf{.[2]}$$

Ερώτημα Ι

Σε αυτό το ερώτημα θα επιλύσουμε αριθμητικά την ροή Couette, με μια ρητή και μια πεπλεγμένη μέθοδο ενσωματώνοντας και την έννοια της χαλάρωσης.

Ρητή μέθοδος

Η εξίσωση. [1], ως παραβολική μερική διαφορική εξίσωση του χρόνου, μας παραπέμπει σε marching-solution,

Παίρνοντας την αδιαστατοποιημένη εξίσωση που δόθηκε προηγουμένως $\frac{\partial u}{\partial t} = \frac{1}{Re_d} \frac{\partial^2 u}{\partial y^2}$ και χρησιμοποιώντας τον ορισμό της Ρητής προσέγγισης όπως αυτός δίνεται στην θεωρία, προκύπτει:

Για το $\frac{\partial u}{\partial t}$ ως κεντρική διακριτοποιημένη έκφραση. Δηλαδή: $\frac{\partial u}{\partial t} = \frac{u_{j+1} - u_{j-1}}{2\Delta t}$

Για το $\frac{\partial^2 u}{\partial y^2}$ ως κεντρική διακριτοπ. β' βαθμού έχουμε: $\frac{\partial^2 u}{\partial y^2} = \frac{u_{j+1} - 2u_j - u_{j-1}}{\Delta y^2}$ **[3]**

Άρα τελικά η αδιαστατοποιημένη εξίσωση παίρνει την μορφή: $\frac{u_{j+1} - u_{j-1}}{2\Delta t} = \frac{1}{Re_d} \frac{u_{j+1} - 2u_j - u_{j-1}}{\Delta y^2} \Rightarrow$

$$\mathrm{A} \cdot u_{j+1} + \mathrm{B} \cdot u_{j-1} + \mathrm{C} \cdot u_j = 0$$

Όπου: $\mathrm{A} = (\frac{1}{2\Delta t} - \frac{1}{Re_d \Delta y^2})$, $\mathrm{B} = (-\frac{1}{2\Delta t} - \frac{1}{Re_d \Delta y^2})$, $\mathrm{C} = \frac{2}{\Delta y^2 Re_d}$

Τελικά, επιλύουμε ως προς $\cdot u_{j+1}$ και

Λαμβάνοντας υπόψη τις τιμές των: u_{j+1}, u_j, u_{j-1}, ως προς τον χρόνο προκύπτει η εξίσωση της ρητής προσέγγισης:

$$\boxed{u_j^{n+1} = u_j^n + a \frac{\Delta t}{\Delta y^2} (u_{j+1}^n - 2u_j^n + u_{j-1}^n)} \quad \textbf{[4]}$$

Η επίλυση ακολουθεί μια βηματική χρονική πορεία, κατά την οποία υπολογίζεται η κατανομή ταχύτητας σε κάθε χρονικό σημείο μέσω του προηγούμενου το οποίο είναι γνωστό, μέχρι να προσεγγιστεί η μόνιμη κατάσταση. Με την μέθοδο αυτήν άρα, έχουμε πάντα άγνωστο στην εξίσωση **[4]** την τιμή $\cdot u_{j+1}$ Για να πραγματοποιηθεί η επίλυση απαιτείται δημιουργία πλέγματος σημείων n, με βήμα 1/n στο μεταξύ τους.

Πεπλεγμένη μέθοδος

Η πεπλεγμένη μέθοδος, προσεγγίζει την επίλυση του φαινομένου με διαφορετικό τρόπο από την ρητή. Συγκεκριμένα, η μερική διαφορική εξίσωση στο δεξί μέλος της **[2]**, προσεγγίζεται αυτή την φορά ως μέσες τιμές μεταξύ του χρονικού βήματος n και n+1. Η διαφοροποίηση αυτή, ονομάζεται μέθοδος Crank-Nicolson και ένα από τα πλεονεκτήματά της, όπως θα δούμε και στην συνέχεια, είναι ότι είναι απόλυτα ευσταθής, αν και δύσκολη να προγραμματιστεί. Η μέθοδος αποδεικνύεται ως εξής:

Έχουμε την εξίσωση:

$\frac{u_j^{n+1}-u_j^n}{\Delta t} = \frac{a}{\Delta y^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$ όπου στην περίπτωση μας έχουμε a=$\frac{1}{2}Re_d$,

Χρησιμοποιώντας τη μέθοδο Crack-Nickolson καταλήγουμε στην εξίσωση:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = \frac{1}{Re_d}\frac{1/2\,(u_{j+1}^{n+1}+u_{j+1}^n-2u_j^n-2u_j^{n+1}+u_{j-1}^n)}{\Delta y^2} \qquad \textbf{[5]}$$

Ομαδοποιώντας την εξίσωση λοιπόν στο δεξί μέλος με n+1 και το αριστερό με n, ως προς τον χρόνο δηλαδή, έχουμε:

$$\left(-\frac{\Delta t}{2(\Delta y)^2 Re_d}\right)u_{j-1}^{n+1} + \left(1+\frac{\Delta t}{(\Delta y)^2 Re_d}\right)u_j^{n+1} + \left(-\frac{\Delta t}{2(\Delta y)^2 Re_d}\right)u_{j+1}^{n+1} = \left(1-\frac{\Delta t}{(\Delta y)^2 Re_d}\right)u_j^n + \left(\frac{\Delta t}{2(\Delta y)^2 Re_d}\right)(u_{j-1}^n + u_{j+1}^n) \quad \textbf{[6]}$$

Άρα καταλήγουμε στην εξίσωση: $\mathrm{A}\cdot u_{j-1}^{n+1} + \mathrm{B}\cdot u_j^{n+1} + \mathrm{A}\cdot u_{j+1}^{n+1} = \mathrm{K}_j$, **[7]**

Όπου: $\mathrm{A} = -\frac{\Delta t}{2(\Delta y)^2 Re_d}$, **[8]**

$\mathrm{B} = 1 + \frac{\Delta t}{(\Delta y)^2 Re_d}$, **[9]**

$\mathrm{K_j} = \left(1-\frac{\Delta t}{(\Delta y)^2 Re_d}\right)u_j^n + \frac{\Delta t}{2(\Delta y)^2 Re_d}(u_{j+1}^n + u_{j-1}^n)$ **[10]**

$\Delta y = \frac{D}{N}$ (D είναι η κάθετη απόσταση μεταξύ των πλακών και N είναι ο συνολικός αριθμός των των timestep).

Οριακές συνθήκες

α) Η ταχύτητα πάνω στο ακίνητο τοίχωμα είναι: $u_1 = 0$ (Συνθήκη Neumann)

β) Η ταχύτητα πάνω στο κινούμενο πίνακα είναι: $u_{N+1} = 1$ (Συνθήκη Neumann)

γ) Επιπλέον, εφόσον ισχύει ότι $u_{N+1} = 1$ στο τοίχωμα, θα επίσης ότι: $\mathrm{A}\cdot u_{N+1}^{n+1} + \mathrm{B}\cdot u_N^{n+1} = \mathrm{K}_N + \mathrm{A}\cdot u_e$

Έπειτα επειδή έχουμε ένα σύστημα με N-1 εξισώσεις με N-1 αγνώστους, η παραπάνω εξίσωση μπορεί να γραφεί ως:

$$\boxed{Au_1^{n+1} + Bu_2^{n+1} + Au_3^{n+1} = K_2} \qquad \textbf{[11]}$$

Αυτό λόγω τον οριακών συνθηκών. Τελικά, χρησιμοποιώντας την μέθοδο Crack-Nickolson, έχουμε το παρακάτω σύστημα το οποίο έχει τριδιαγώνια μορφή. Για την επίλυση του χρησιμοποιείται ο αλγόριθμος Thomas.

$$\begin{bmatrix} B & A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & B & A & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & A & B & A & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A & B & A & 0 & 0 & 0 & 0 \\ & & & \cdots & & & & & \\ & & & & \cdots & & & & \\ & & & & & \cdots & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & A & B & A \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A & B \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \\ u_5^{n+1} \\ \cdot \\ \cdot \\ \cdot \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{bmatrix} = \begin{bmatrix} K_2 \\ K_3 \\ K_4 \\ K_5 \\ \cdot \\ \cdot \\ \cdot \\ K_{N-1} \\ K_N - Au_e \end{bmatrix}$$

Εφόσον λοιπόν χρησιμοποιήσουμε τον αλγόριθμο Thomas για το παραπάνω σύστημα έχουμε λύση για τις ταχύτητες: $u_2^{n+1}, u_3^{n+1}, u_4^{n+1} \dots \dots u_n^{n+1}$, οι οποίες είναι για timestep: n+1. Επαναλαμβάνουμε την διαδικασία για όλο τον αριθμό των timestep έως ότου η ταχύτητα φτάσει την μόνιμη κατάσταση (steady state) όπως στην εικόνα στην προηγούμενη σελίδα.

Χαλάρωση

Στην περίπτωση της υποχαλάρωσης- υπερχαλάρωσης μπορούμε να αξιοποιήσουμε τις πληροφορίες από την μέχρι τώρα συμπεριφορά της σύγκλισης. Συγκεκριμένα μπορούμε μέσω ενός συντελεστή βαρύτητας ω, να αξιοποιήσουμε την τιμή του $u_{i,j}^n$ και να την συμπεριλάβουμε στον υπολογισμό του $u_{i,j}^{n+1}$. Η τιμή που θα δώσουμε στο ω δείχνει την βαρύτητα που θα δώσουμε στην τιμή της προηγούμενης επανάληψης.

Υποχαλάρωση έχουμε για $0<\omega<1$, όπου θυσιάζουμε ταχύτητα σύγκλισης, αλλά ο αλγόριθμος μας γίνεται πιο ευσταθής.

Το ακριβώς αντίθετο συμβαίνει στην υπερχαλάρωση. Δηλαδή με την υπερχαλάρωση πετυχαίνουμε έναν πολύ μικρό χρόνο υπολογισμού χωρίς όμως να επιτυγχάνεται ο βαθμός ευστάθειας σε σχέση με την υποχαλάρωση.

Η ροϊκή εξίσωση, συμπεριλαμβάνοντας και τον συντελεστή βαρύτητας ω, γίνεται:

$$u_j^{n+1} = \omega \cdot \left[u_j^n + a\frac{\Delta t}{\Delta y^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)\right] + (1-\omega)u_j^n \quad \textbf{[12]}$$

Ερώτημα ΙΙ

Σε αυτό το ερώτημα, θα αναλύσουμε την ευστάθεια των δύο μεθόδων με βάση το χρονικό βήμα που θα επιλέξουμε και με βάση τον συντελεστή χαλάρωσης, ενώ θα γίνει ταυτόχρονα και διερεύνηση για τις τιμές που θα επιλέξουμε για την επίλυση του προβλήματός μας.

Ευστάθεια

Για την ευστάθεια της διαφορικής εξίσωσης χρησιμοποιήθηκε η μέθοδος Von Neumann(stability method). Η ανάλυση ευστάθειας βασίστηκε στην μέθοδο Von Neumann (Von Neumann stability method).

Για τη ρητή μέθοδο, το κριτήριο που προκύπτει είναι : $\boldsymbol{\Delta t\, Re*(\Delta y)\, 2 \leq 1\, 2}$

Ενώ για την πεπλεγμένη , θα υπολογίσουμε το Δt = E $*$ Re $(\Delta y)^2$. Εφόσον η μέθοδος Crank-Nicolson είναι εξ ορισμού ευσταθής το E δεν έχει περιορισμό στην τιμή του, όπως θα δούμε και παρακάτω.

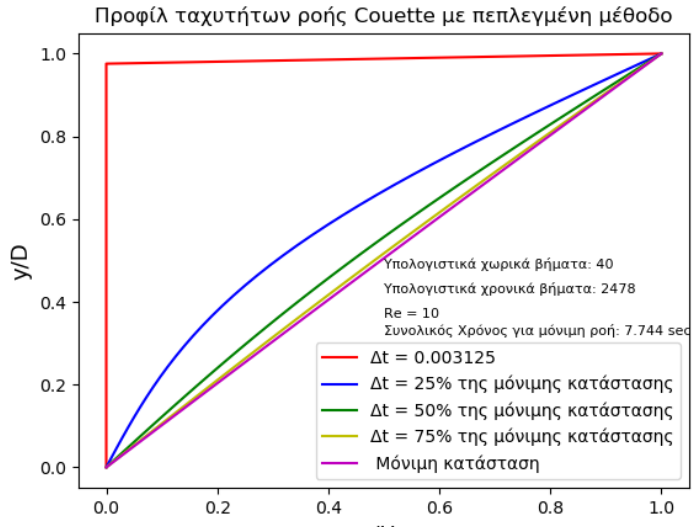
Ρητή μέθοδος

ii) Συντελεστής Χαλάρωσης

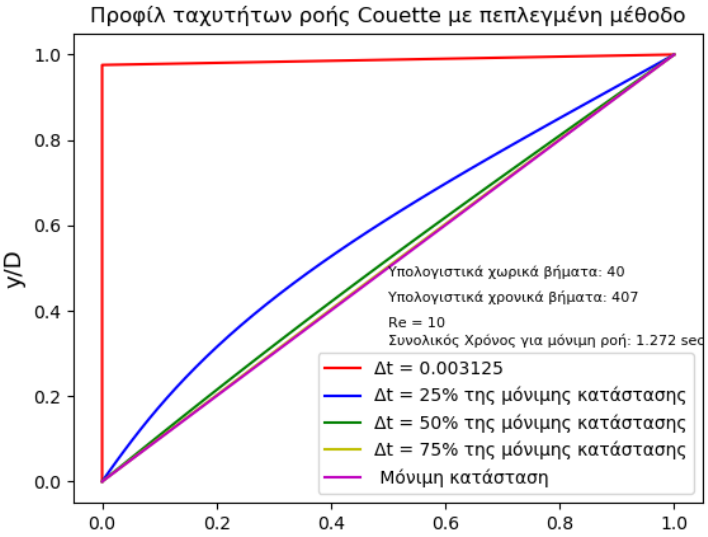
Έγινε μια διερεύνηση των τιμών του συντελεστή χαλάρωσης που απαιτείται για την επίλυσή μας, κρατώντας σταθερές τις υπόλοιπες μεταβλητές που το διέπουν.

Παρακάτω παρατίθεται κάποια ενδεικτικά διαγράμματα ροής με στόχο την απεικόνιση της επίδρασης τόσο των timestep, όσο και της χαλάρωσης υποχαλάρωση-υπερχαλάρωση.

Σχόλια: Σε αυτήν την περίπτωση χρησιμοποιήσαμε συντ. ω=0.5 για Re=10, Δy=40
Αυτό, διότι παρατηρήθηκε πολύ αξιόλογη ευστάθεια για ω=0.5 και σε ικανοποιητικό χρόνο διεκπεραίωσης της λύσης (αν βάζαμε αρκετά μικρότερο ω θα είχαμε θέμα στον χρόνο επίλυσης).



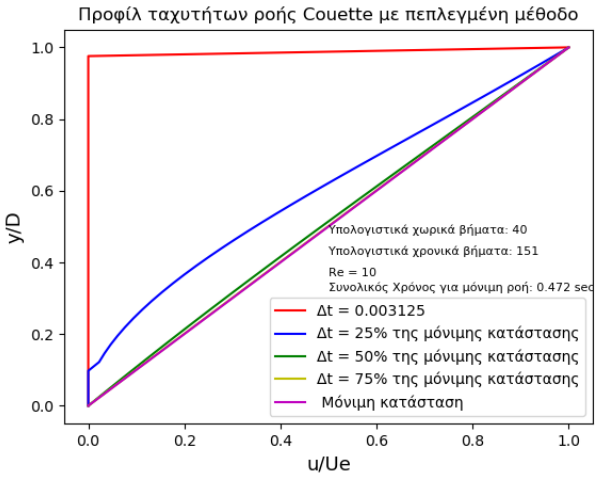
Διάγραμμα 1



Διάγραμμα 2

Σχόλια: Σε αυτήν την περίπτωση χρησιμοποιήσαμε συντ. ω=1.5 για Re=10, Δy=40
Μπορούμε εύκολα λοιπόν εύκολα να βγάλουμε συμπεράσματα για την σύγκριση υποχαλάρωσης-υπερχαλάρωσης παρατηρώντας τον χρόνο που χρειάστηκε για την μόνιμη κατάσταση (1.675 sec) σε αντίθεση με την υποχαλάρωση που ήθελε (11.491 sec). Αλλά και για την ευστάθεια όπου η παραπάνω είναι πιο κοντά στην θεωρητική ροή Couette.

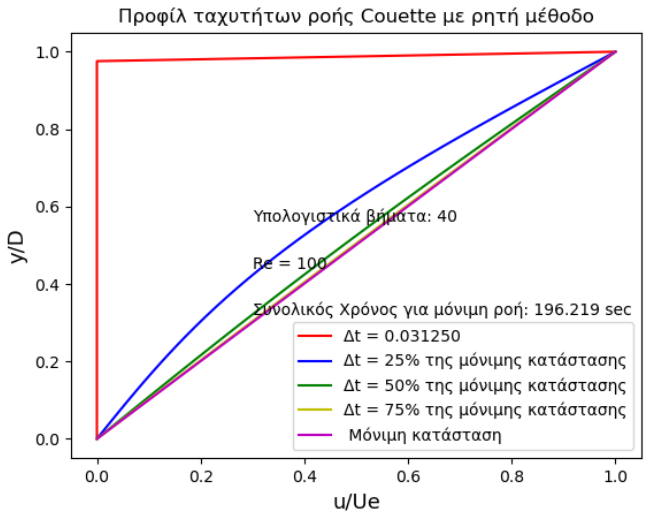
Σχόλια: Σε αυτήν την περίπτωση χρησιμοποιήσαμε συντ. ω=1.8 για Re=10, Δy=40
Μπορούμε εύκολα λοιπόν εύκολα να βγάλουμε συμπεράσματα ότι υπάρχει ένα πλατό μέχρι το οποίο μπορεί να μας βοηθήσει η υπερχαλάρωση, όπως παρατηρείται στο διάγραμμα 3, έχει ισορροπήσει σε μόνιμη ροή το σύστημα, αλλά οι λύσεις δεν είναι τόσο ακριβείς και η όλη επίλυση καθίσταται αναξιόπιστη.

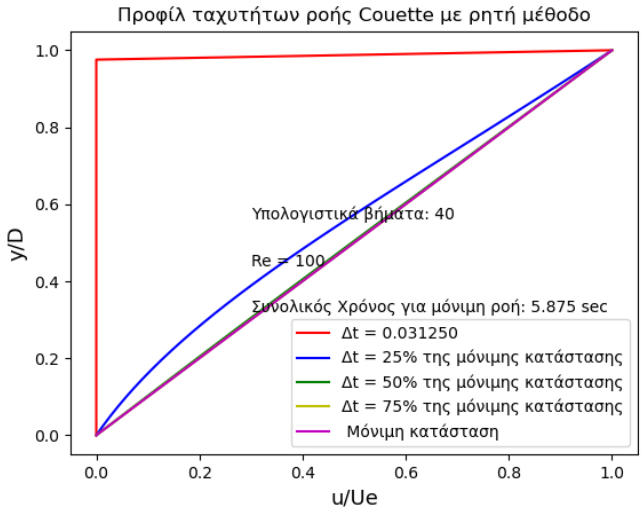


Διάγραμμα 3

Γίνεται και η αντίστοιχη μελέτη για Re = 100.

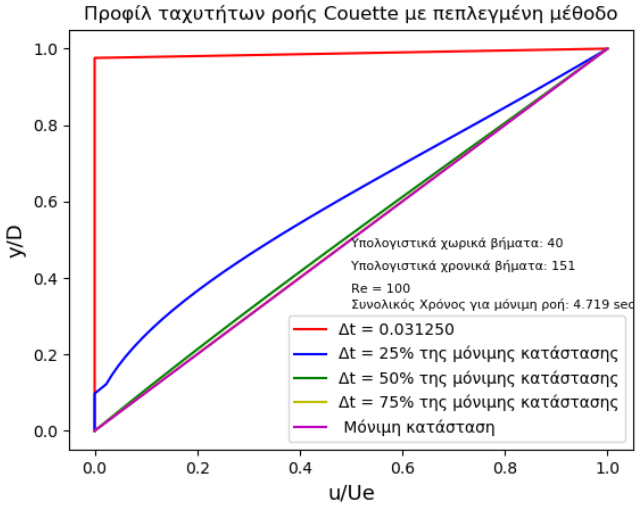
Σχόλια: Σε αυτήν την περίπτωση χρησιμοποιήσαμε συντ. ω=0.3 για Re=100, Δy=40
Δοκιμάζοντας έναν ω=0.3 λάβαμε με μεγαλύτερο Re, λαμβάνουμε μια ικανοποιητική λύση όσον αφορά την ευστάθεια όπως αναμενόταν.





Σχόλια: Σε αυτήν την περίπτωση χρησιμοποιήσαμε συντ. ω=1.5 για Re=100, Δy=40 , ως ενδιάμεση κατάσταση ενώ,

Σχόλια: Σε αυτήν την περίπτωση χρησιμοποιήσαμε συντ. ω=1.8 για Re=100, Δy=40
Συγκρίνοντας με το παραπάνω διάγραμμα προκύπτουν τα ίδια συμπεράσματα όπως και την σύγκριση για Re=10. Αξίζει να σημειωθεί ότι ο κώδικας που αναπτύξαμε στο υπολογιστικό περιβάλλον της Python δουλεύει ως ω=1.8 θεωρώντας το max. για την υπερχαλάρωση. Για μεγαλύτερα ω ο κώδικας κάνει blow up!



Πεπλεγμένη Μέθοδος

Όπως αναφέρθηκε και προηγουμένως, η πεπλεγμένη μέθοδος δίνει αρκετά μεγαλύτερη ευστάθεια στο σύστημα μας συγκριτικά με την ρητή μέθοδο. Στον κώδικα που αναπτύξαμε έχουμε συμπεριλάβει μια μεταβλητή Ε, η οποία λειτουργεί,

<u>i) Με βάση το timestep:</u>

Ο υπολογισμός του βήματος και σε αυτήν την περίπτωση λαμβάνεται από την βιβλιογραφία και συγκεκριμένα ως:

$$\Delta t = E \cdot Re_d \cdot \Delta y^2$$ **.[13]**

Αφού η τεχνική Cranck-Nickolson δίνει ευστάθεια υπό οποιαδήποτε συνθήκη, εισάγουμε μια παράμετρο «Ε» η οποία λαμβάνει οποιαδήποτε τιμή μεταξύ του διαστήματος 1<E<4000. Ο ορισμός της γίνεται ως εξής: $\mathrm{E} = \frac{\Delta t}{Re_d \cdot (\Delta y)^2}$.

Επομένως οι πίνακες A,B,K μπορούν να γραφούν:

$$\mathrm{A} = -\frac{E}{2}, \quad \mathrm{B} = 1 + E, \quad \mathrm{K_j} = (1 - E)u_j^n + \frac{E}{2}(u_{j+1}^n + u_{j-1}^n) \, .$$

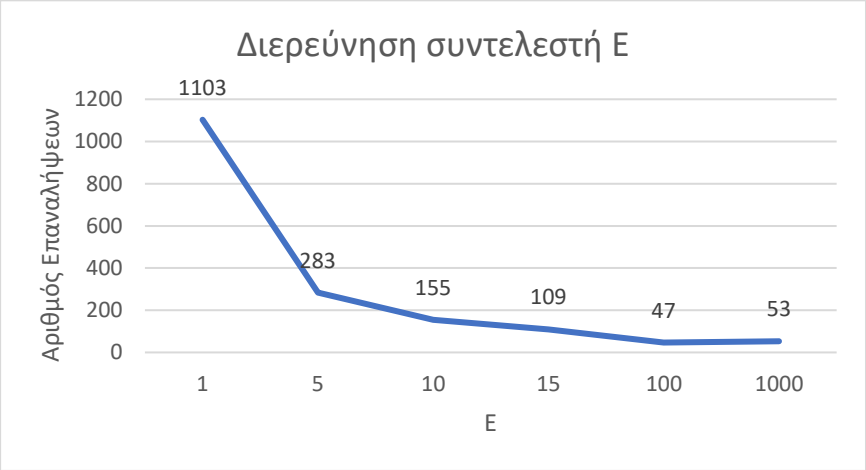
Από τα παραπάνω, προκύπτει, η λύση της ροής σε μόνιμη κατάσταση είναι ανεξάρτητη του αριθμού Re_d

Για τον προσέγγιση της βέλτιστης τιμής για την επίλυση του προβλήματός μας, έγινε μια διερεύνηση της τιμής αυτής. Η διερεύνηση αυτή, έγινε με παρατήρηση των διαγραμμάτων, αλλά και του αριθμού των επαναλήψεων. Κρατήθηκαν σταθερές οι υπόλοιπες μεταβλητές του προβλήματος καθ' όλη τη διάρκεια της διερεύνησης, δηλαδή: Re = 100, y = 40, error threshold = 10^{-5} και συντελεστής χαλάρωσης ω = 0,9.

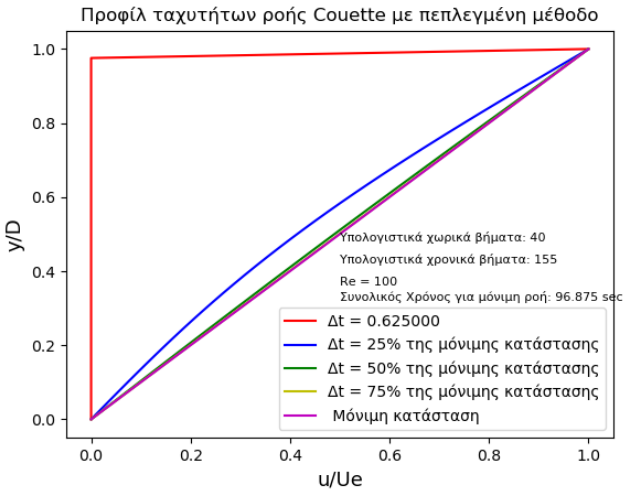
<u>ii) Με βάση τον συντελεστή χαλάρωσης:</u>

Ομοίως με την ρητή.

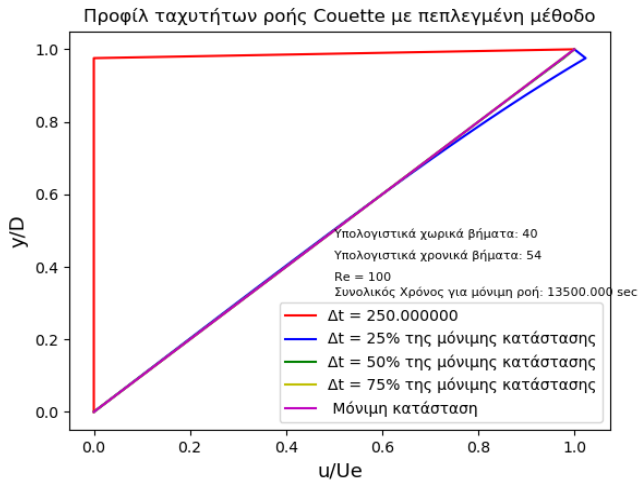
Για τον συντελεστή Ε διεξήχθησαν οι παρακάτω τιμές.



Γράφημα 1



Διάγραμμα 4



Διάγραμμα 5

<u>Σχόλια</u>

Παρατηρήθηκε ότι για μεγάλες τιμές συντελεστή Ε, τα προφίλ ταχυτήτων αποκλίνουν από την τιμή της μόνιμης κατάστασης.

Αυτό γίνεται λόγω του μεγάλου χρονικού βήματος που εισάγεται με τον τρόπο αυτό, με αποτέλεσμα να αυξάνεται κατά πολύ ο χρόνος μέχρι την μόνιμη ροή, (13500sec στο διάγραμμα 5) ενώ ταυτόχρονα χάνονται πολλές ενδιάμεσες πτυχές του φαινομένου.

Ερώτημα III

Για την επίλυση

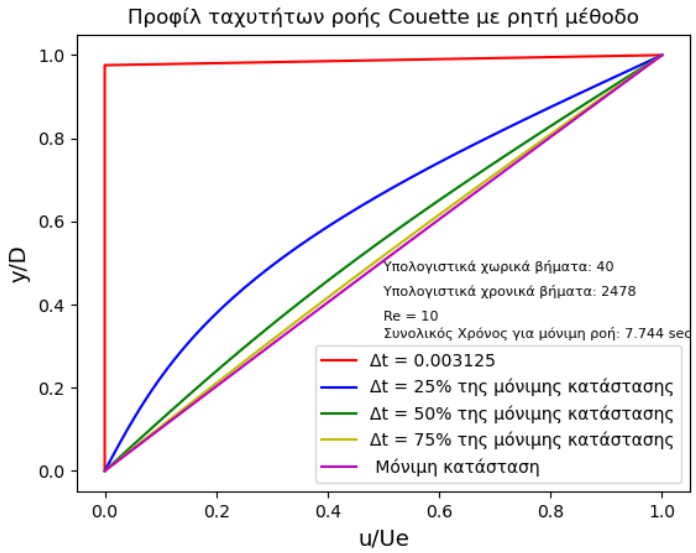
Οι κατανομές ταχυτήτων παρουσιάζονται παρακάτω.

Ρητή Μέθοδος:

Για Δy=40

Για Re_d =10 :

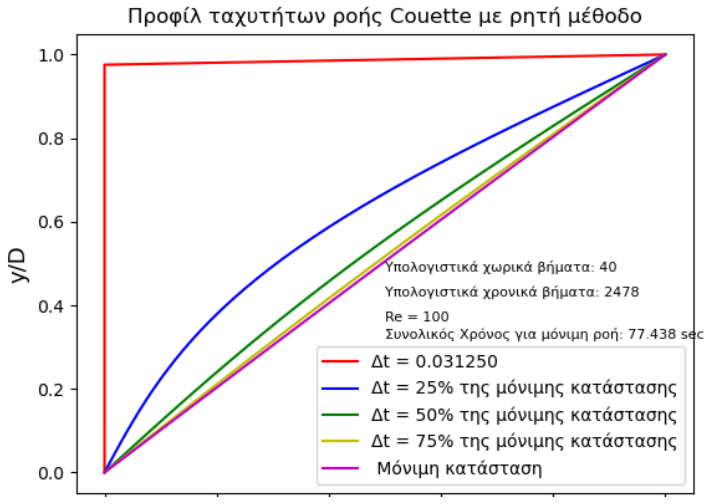
Re	10
Steps-y	40
Δt	0.003125
T_{total}	7,744
ω	0.5
Δy	0.025



Διάγραμμα 6

Για Re_d =100 :

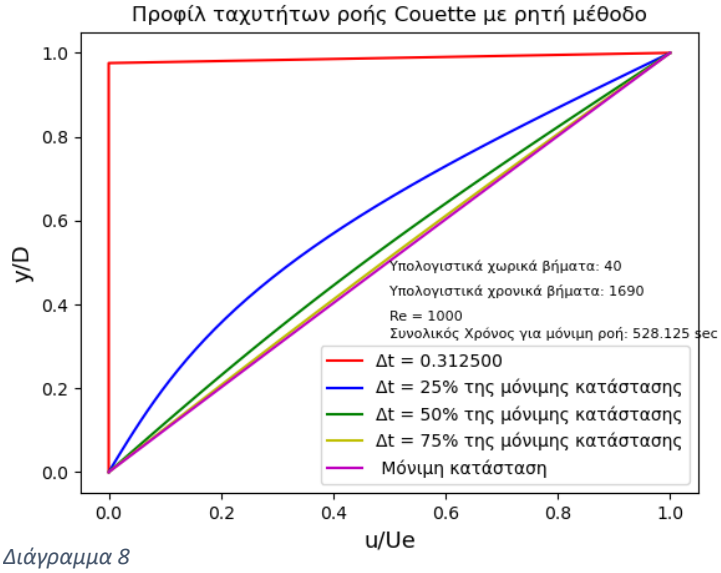
Re	100
Steps-y	40
Δt	0.031250
T_{total}	77.443
ω	0.5
Δy	0.025



Διάγραμμα 7

Για Re_d =1000

Re	1000
Steps-y	40
Δt	0.312500
T_{total}	528.125
ω	0.7
Δy	0.025

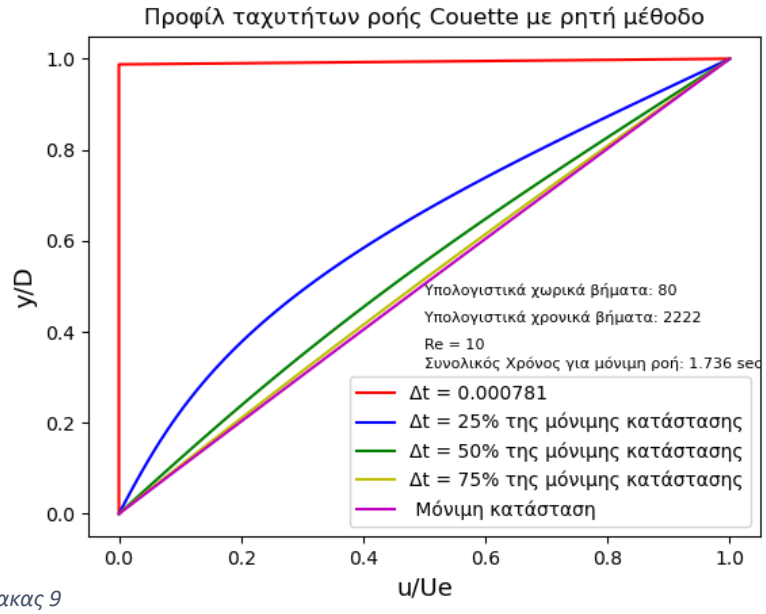


Διάγραμμα 8

Για Δy =80

Για Re_d =10 :

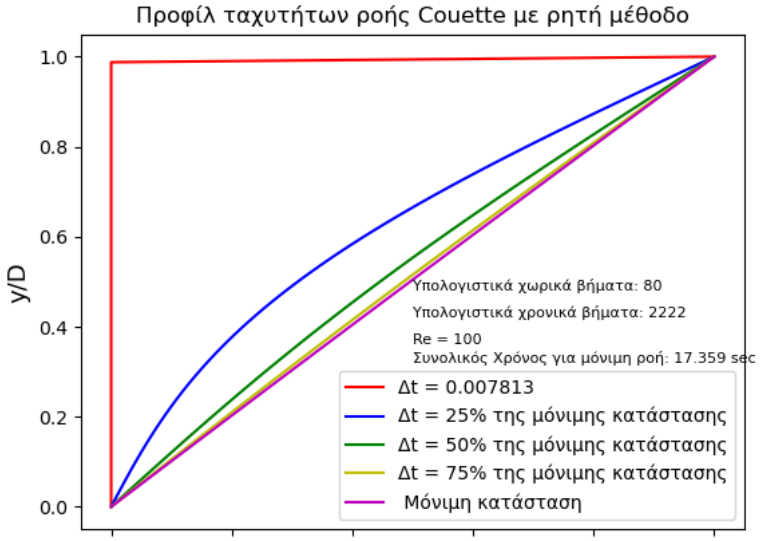
Re	10
Steps-y	80
Δt	0.000781
T_{total}	1.736
ω	1.2
Δy	0.0125



Πίνακας 9

Για Re_d =100 :

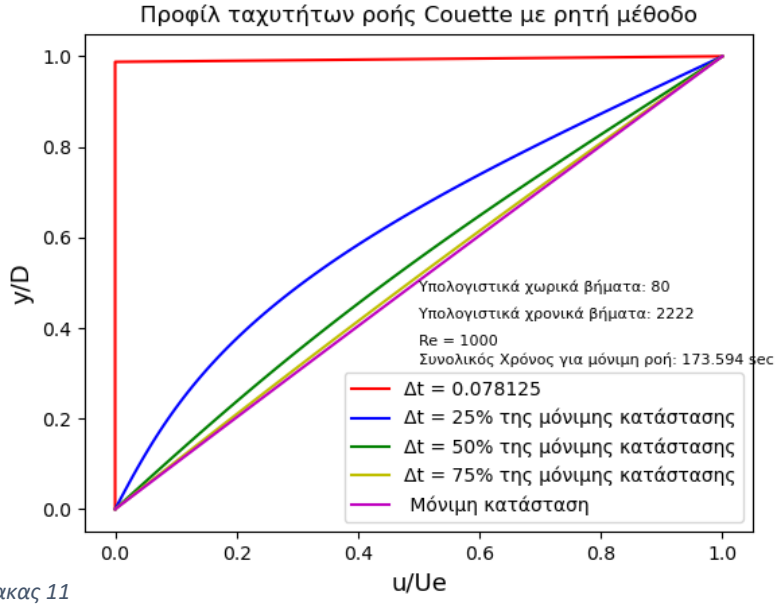
Re	100
Steps-y	80
Δt	0.007813
T_{total}	17.359
ω	1.2
Δy	0.0125



Πίνακας 10

Για Re_d =1000 :

Re	1000
Steps-y	80
Δt	0.0078125
T_{total}	173.594
ω	1.2
Δy	0.0125

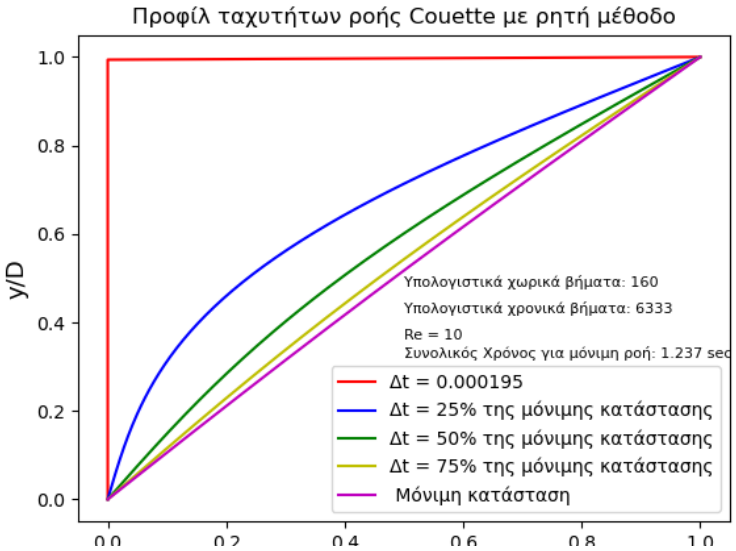


Πίνακας 11

Για Δy=160

Για Re_d =10 :

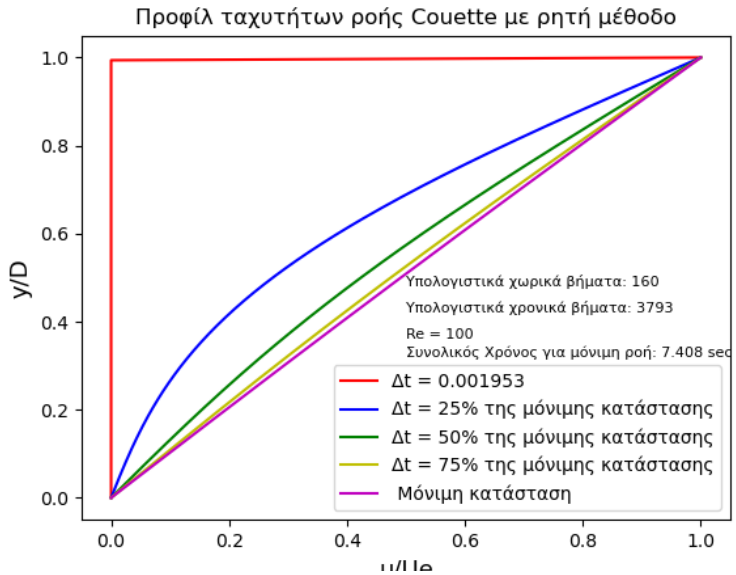
Re	10
Steps-y	160
Δt	0.000195
T_{total}	1.237
ω	1.2
Δy	0.00625



Διάγραμμα 12

Για Re_d =100 :

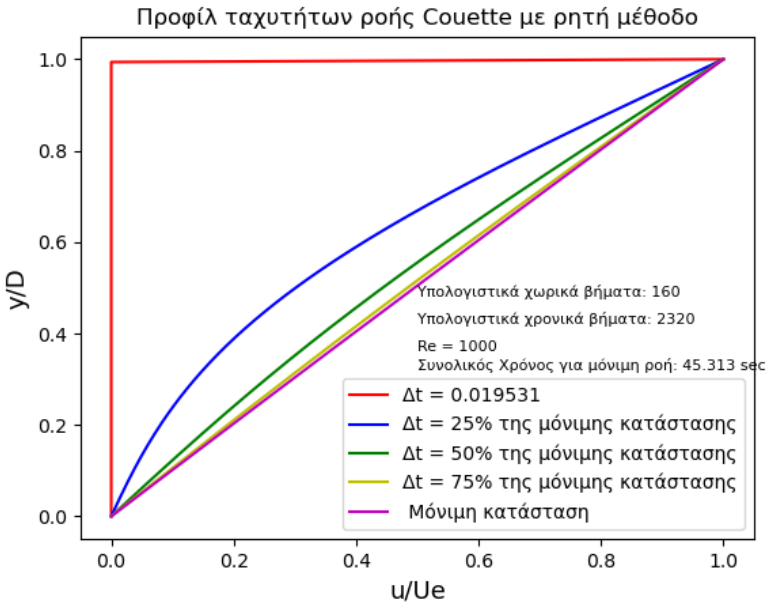
Re	100
Steps-y	160
Δt	0.001953
T_{total}	7.408
ω	1.5
Δy	0.00625



Διάγραμμα 13

Για Re_d =1000 :

Re	1000
Steps-y	160
Δt	0.019531
T_{total}	45.313
ω	1.7
Δy	0.00625



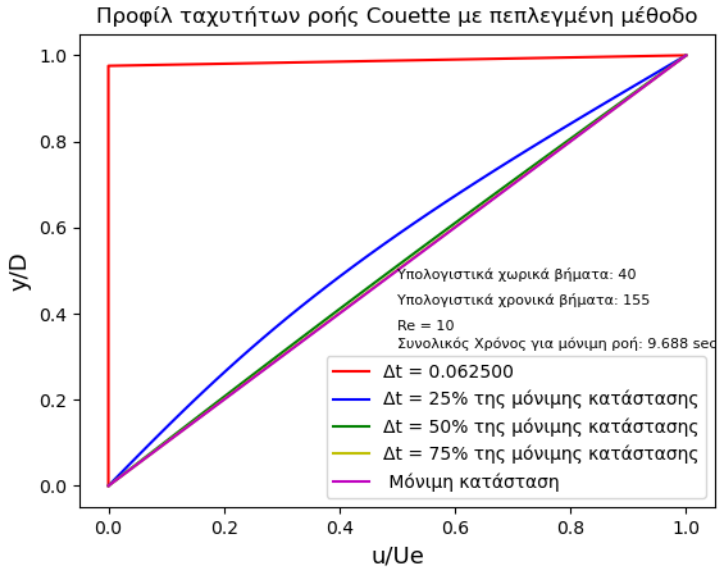
Διάγραμμα 14

Πεπλεγμένη Μέθοδος

Για Δy=40

Για Re_d =10 :

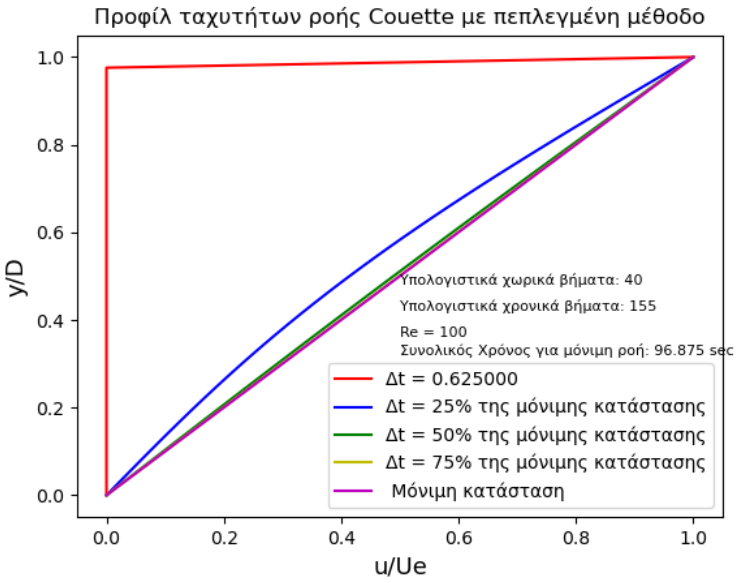
Re	10
Steps-y	40
Δt	0.0625
T_{total}	9.688
ω	0.9
Δy	0.025
E	10



Διάγραμμα 15

Για Re_d =100 :

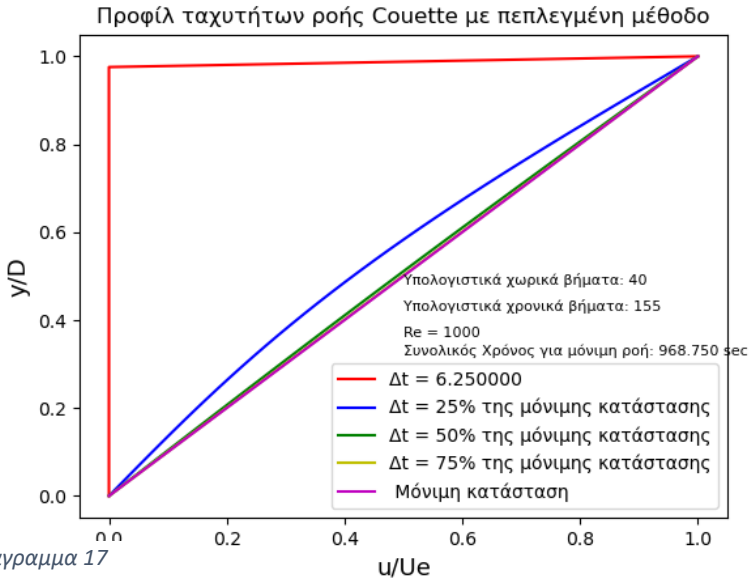
Re	100
Steps-y	40
Δt	0.625
T_{total}	96.875
ω	0.9
Δy	0.025
E	10



Διάγραμμα 16

Για Re_d =1000 :

Re	1000
Steps-y	40
Δt	6.25
T_{total}	968.75
ω	0.9
Δy	0.025
E	10

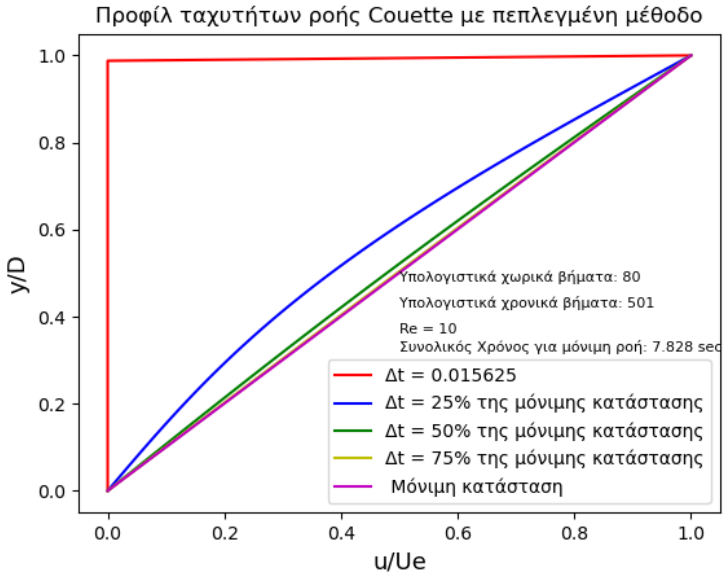


Διάγραμμα 17

Για Δy=80

Για Re_d =10 :

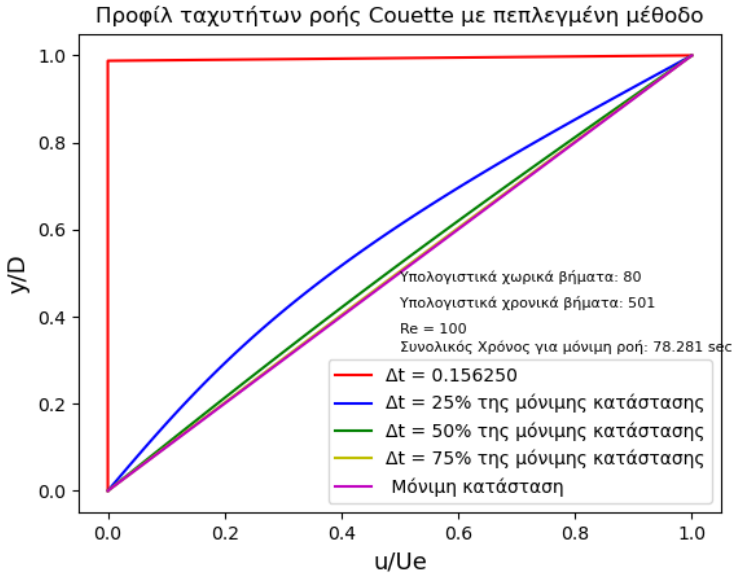
Re	10
Steps-y	80
Δt	0.015625
T_{total}	7.828
ω	0.9
Δy	0.0125
E	15



Διάγραμμα 18

Για Re_d =100 :

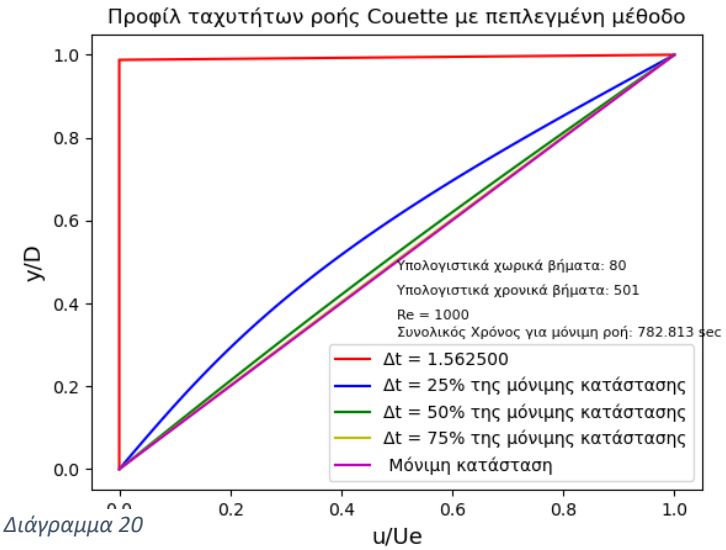
Re	100
Steps-y	80
Δt	0.15625
T_{total}	78.281
ω	0.9
Δy	0.0125
E	15



Διάγραμμα 19

Για Re_d =1000 :

Re	1000
Steps-y	80
Δt	1.5625
T_{total}	782.813
ω	0.9
Δy	0.0125
E	15

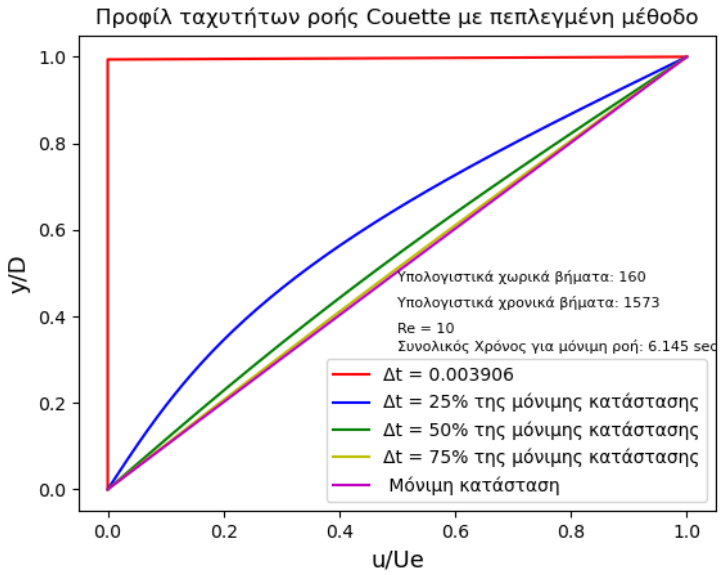


Διάγραμμα 20

Για Δy=160

Για Re_d =10 :

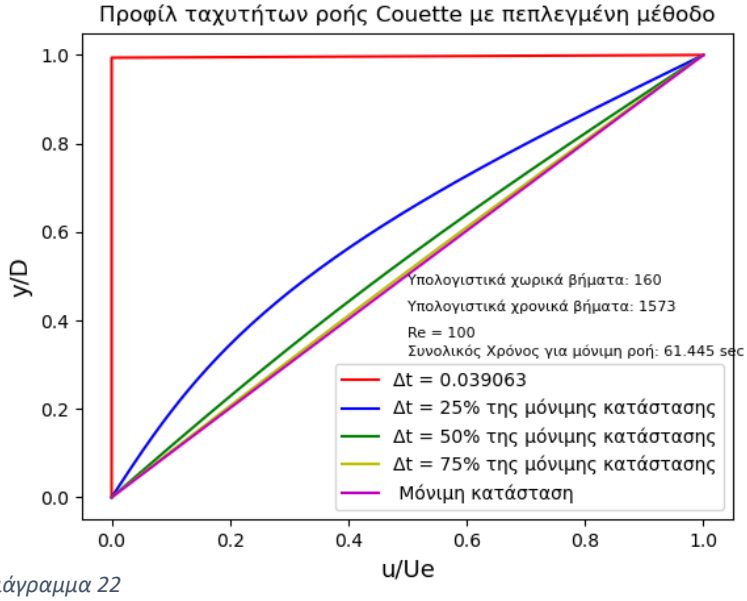
Re	10
Steps-y	160
Δt	0.003906
T_{total}	6.145
ω	0.9
Δy	0.00625
E	10



Διάγραμμα 21

Για Re_d =100 :

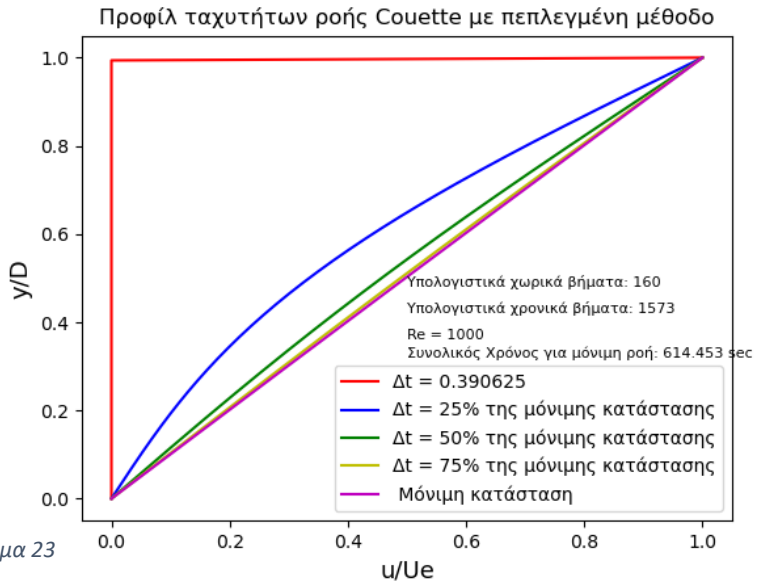
Re	100
Steps-y	160
Δt	0.039063
T_{total}	61.445
ω	0.9
Δy	0.00625
E	10



Διάγραμμα 22

Για Re_d =1000 :

Re	1000
Steps-y	160
Δt	0.390625
T_{total}	614.453
ω	0.9
Δy	0.00625
E	10



Διάγραμμα 23

Ερώτημα IV (συμπεράσματα)

Πολλά από τα στοιχεία που θα κατατεθούν σε αυτό το ερώτημα έχουν ήδη απαντηθεί σαν σχόλια στα προηγούμενα ερωτήματα.

Ρητή μέθοδος:

Το βήμα που εκλέχθηκε κινείται γύρω από τις τιμές του $\frac{1}{2}Re_d(\Delta y)^2$. Το βήμα επηρεάζει την ευστάθεια των υπολογισμών. Από τα παραπάνω διαγράμματα παρατηρούμε ότι όσο αυξάνουμε τις τιμές του Re η ροή απαιτεί περισσότερο χρόνος ώστε να φτάσει την μόνιμη κατάσταση. Επίσης για το step-y, εάν διατηρήσουμε σταθερές τις υπολογιστικές μας συνθήκες, δηλαδή σταθερά Re, Steps-y, ω, τότε παρατηρούμαι ότι για αυξημένο Δy ο χρόνος που απαιτείται για να φτάσουμε στην μόνιμη ροή μειώνεται. Οι κύριες διαφορές μεταξύ των καμπυλών της ροής παρατηρήθηκαν από το 25% ως το 75% της ροής. Επίσης για y-step=160, παρατηρήθηκε πολύ μεγαλύτερος χρόνος σύγκλησης της μεθόδου υπολογισμού σε σχέση με τα μικρότερα Δy. Ο πολύ μεγάλος χρόνος υπολογισμού για y-step=160 σε συνεργασία με την εισαγωγή της έννοιας την χαλάρωσης, μας οδήγησε στην χρησιμοποίηση της υπερχαλάρωσης για μεγαλύτερα Δy Επίσης όπως αναμενόταν από την θεωρία, η ρητή μέθοδος αποτελεί λιγότερο ευσταθής σε σχέση με την πεπλεγμένη.

Πεπλεγμένη μέθοδος:

Η πεπλεγμένη μέθοδος όπως παρατηρείται μας δίνει ευστάθεια ανεξαρτήτως παραγόντων. Αποτελεί μια πιο πολύπλοκη τεχνική σε σχέση με την πιο απλοϊκή ρητή, όμως μπορούμε να είμαστε πιο σίγουροι για την ευστάθεια της. Επίσης παρατηρούμαι ότι δια ίδιο y-step ο χρόνος μέχρι την μόνιμη ροή για αυξανόμενο Re αυξάνεται και αυτός κάτι το οποίο ήταν αναμενόμενο (περίπου όσο πολλαπλασιάζεται ο Re, π.χ. για y-step=40 και Re=10 ο χρόνος ήταν 9.688 sec ενώ για Re=100 και ίδιο y-step ο χρόνος είναι 98.875 sec, όμοια συμπεριφορά και για Re=1000). Επιπλέον, για ίδιο αριθμό Re και αυξανόμενο y-step ο χρόνος μέχρι την μόνιμη ροή παρατηρείται ότι μειώνεται (π.χ. για Re=100 και y-step=40 ο χρόνος είναι 96.875 sec, για y-step=80 ο χρόνος είναι 78.281 sec και για y-step=160 ο χρόνος είναι 61.445 sec).

Χαλάρωση:

Όπως είχε αναφερθεί για την χαλάρωση δοκιμάσαμε διάφορους συντελεστές ω ώστε να λάβουμε τα επιθυμητά συμπεράσματα. Πράγματι όσο πλησιάζαμε το ω στο 0 κατά τους υπολογισμούς που επιδιώξαμε παρατηρήσαμε πολύ μεγαλύτερο χρόνο υπολογισμού αλλά παράλληλα μεγαλύτερη ευστάθεια. Όσο αυξάναμε το ω χάναμε από ευστάθεια κερδίζοντας όμως χρόνο υπολογισμού. Όπως είχε αναφερθεί για ω=1 δεν υφίσταται ευστάθεια κάτι το οποίο φαίνεται στον τύπο της ευστάθειας. Για την ρητή μέθοδο, μας επιτράπηκε να χρησιμοποιήσουμε ως ω=1.8 (max) αφού για μεγαλύτερο ω εμφανιζόταν κάποια ροϊκά φαινόμενα τα οποία δεν ήταν σωστά (ω=1.9) και για ακόμα μεγαλύτερα ω ο κώδικας έκανε blow up και εμφανίζει αστάθεια. Για την πεπλεγμένη μέθοδο, ο συντελεστής ω που χρησιμοποιηθηκε,

Βιβλιογραφία

1) Fundamentals of fluid mechanics, Bruce R. Munson, Alric P. Rothmaye, Theodore H. Okiishi, Wade W. Huebsch

2) Διδακτικές σημειώσεις μαθήματος: Υπολογιστική Ρευστομηχανική

3) Computational fluid mechanics, John D. Anderson Jr., Mc-Graw Hills series

Code listing

```
import numpy as np
import matplotlib.pyplot as plt
import time as ti
from tkinter import messagebox


def riti(y, Re, sfalma, omega, tropos):
    dy = 1 / y
    dt = 0.5 * Re * dy ** 2
    y_tot = np.linspace(0, 1, y + 2)
    n = 1
    u = np.zeros(len(y_tot))
    u[-1] = 1
    logic = True
    velocity_profiles = u.copy()
    plt.figure(1)
    u_avg = np.zeros(len(y_tot))
    u_prev = np.zeros(len(y_tot))
    while logic:
        for j in range(1, len(y_tot) - 1):
            if tropos == 1:
                u_avg[j] = (1 - (2 * dt / (Re * dy ** 2))) * u[j] + (dt / (Re * (dy ** 2))) * (u_prev[j + 1] + u[j - 1])
                u[j] = u_prev[j] + omega * (u_avg[j] - u_prev[j])
            else:
                u[n, j] = u[n - 1, j] + (dt / (Re * (dy ** 2))) * (u[n - 1, j + 1] - 2 * u[n - 1, j] + u[n - 1, j - 1])
        sigklisi = np.abs(u_prev[:] - u[:])
        u_prev = u.copy()
        if all(sigklisi <= sfalma):
            logic = False
        n = n + 1
        velocity_profiles = np.vstack([velocity_profiles, u])
    n = n - 1
    time = dt * n
    plt.plot(velocity_profiles[0], y_tot, color='r', label='Δt = %f ' % dt)
    plt.plot(velocity_profiles[int(0.25 * n)], y_tot, color='b', label='Δt = 25% της μόνιμης κατάστασης')
    plt.plot(velocity_profiles[int(0.5 * n)], y_tot, color='g', label='Δt = 50% της μόνιμης κατάστασης')
    plt.plot(velocity_profiles[int(0.75 * n)], y_tot, color='y', label='Δt = 75% της μόνιμης κατάστασης')
    plt.plot(velocity_profiles[int(n)], y_tot, color='m', label=' Μόνιμη κατάσταση')
    plt.title('Προφίλ ταχυτήτων ροής Couette με ρητή μέθοδο')
    plt.xlabel('u/Ue', fontsize=13)
    plt.ylabel('y/D', fontsize=13)
    plt.text(0.5, 0.32, 'Συνολικός Χρόνος για μόνιμη ροή: %.3f' % time + ' sec', fontsize=8)
    plt.text(0.5, 0.36, 'Re = %.0f' % Re, fontsize=8)
    plt.text(0.5, 0.42, 'Υπολογιστικά χρονικά βήματα: %.0f' % n, fontsize=8)
    plt.text(0.5, 0.48, 'Υπολογιστικά χωρικά βήματα: %.0f' % y, fontsize=8)
    plt.legend()
    plt.show()


def peplegmeni(y, Re, E, sfalma, omega, tropos):
    dy = 1 / (y)
    dt = E * Re * (dy) ** 2
    y_tot = np.linspace(0, 1, y + 2)
    u = np.zeros(len(y_tot))
    u[-1] = 1
    logic = True
    n = 0
    velocity_profiles = u.copy()
    while logic:
        u_prev = u.copy()
        K = np.zeros(y)
        n = n + 1
        A = -(E / 2) * np.ones(y)
```

```
        B = (1 + E) * np.ones(y)
        C = -(E / 2) * np.ones(y)
        for j in range(1, len(y_tot) - 1):
            K[j - 1] = (1 - E) * u_prev[j] + E/2 * (u_prev[j + 1] + u_prev[j - 1])
        K[-1] = K[-1] - C[-1] * u[-1]
    #Thomas
        for i in range(1, len(A)):
            mc = A[i] / B[i - 1]
            B[i] = B[i] - mc * C[i - 1]
            K[i] = K[i] - mc * K[i - 1]
        Thomas = A.copy()
        Thomas[-1] = K[-1] / B[-1]
        for j in range(len(A)-2, -1, -1):
            Thomas[j] = (K[j] - C[j] * Thomas[j + 1]) / B[j]
        for j in range(1, len(y_tot) - 1):
            if tropos == 1:
                u[j] = (omega * Thomas[j - 1]) + (1 - omega) * u_prev[j]
            else:
                u[j] = Thomas[j - 1]
        velocity_profiles = np.vstack([velocity_profiles, u])
        sigkrisi = abs(u - u_prev)
        if all(sigkrisi <= sfalma):
            logic = False
    plt.figure(2)
    time = n * dt
    plt.plot(velocity_profiles[0], y_tot, color='r', label='Δt = %f ' % dt)
    plt.plot(velocity_profiles[int(0.25 * n)], y_tot, color='b', label='Δt = 25% της μόνιμης κατάστασης')
    plt.plot(velocity_profiles[int(0.5 * n)], y_tot, color='g', label='Δt = 50% της μόνιμης κατάστασης')
    plt.plot(velocity_profiles[int(0.75 * n)], y_tot, color='y', label='Δt = 75% της μόνιμης κατάστασης')
    plt.plot(velocity_profiles[int(n)], y_tot, color='m', label=' Μόνιμη κατάσταση')
    plt.title('Προφίλ ταχυτήτων ροής Couette με πεπλεγμένη μέθοδο')
    plt.xlabel('u/Ue', fontsize=13)
    plt.ylabel('y/D', fontsize=13)
    plt.text(0.5, 0.32, 'Συνολικός Χρόνος για μόνιμη ροή: %.3f' % time + ' sec', fontsize=8)
    plt.text(0.5, 0.36, 'Re = %.0f' % Re, fontsize=8)
    plt.text(0.5, 0.42, 'Υπολογιστικά χρονικά βήματα: %.0f' % n, fontsize=8)
    plt.text(0.5, 0.48, 'Υπολογιστικά χωρικά βήματα: %.0f' % y, fontsize=8)
    plt.legend()
    plt.show()


control = 0
while control == 0:
    option = float(input("Define solving method [1 for explicid, 2 for implicid]: "))
    if option == 1 or option == 2:
        control = 1
    else:
        messagebox.showinfo("Wrong input", "Please define a proper solving method (1 or 2)")
if option == 1:
    control = 0
    while control == 0:
        y = int(input("Define number of grid points: [>1]"))
        if y > 1:
            control = 1
        else:
            messagebox.showinfo("Wrong input", "Please define a proper number of grid points: ")
    control = 0
    while control == 0:
        Re = float(input("Define Reynolds number [>0]"))
        if Re > 0 :
            control = 1
        else:
            messagebox.showinfo("Wrong input", "Please define a proper number : ")
    control = 0
    while control == 0:
        error = float(input("Define desired error threshold: [<1]"))
```

```
    if 1 > error > 0:
      control = 1
    else:
      messagebox.showinfo("Wrong input", "Please define a proper error threshold: [<1]")
  control = 0
  while control == 0:
    relax = input("Do u wish to add relaxation method?[y, n]")
    if relax == 'y':
      tropos = 1
      control = 1
      control2 = 0
      while control2 == 0:
        omega = float(input("Define desired omega factor: [0 < ω < 2]"))
        if 2 > omega > 0:
          control2 = 1
        else:
          messagebox.showinfo("Wrong input", "Please define a proper value: [0 < ω < 2]")
    elif relax == 'n':
      tropos = 0
      control = 1
    else:
      messagebox.showinfo("Wrong input", "Please give a proper answer: [y, n]")
  riti(y, Re, error, omega, tropos)
else:
  control = 0
  while control == 0:
    y = int(input("Define number of grid points: [>1]"))
    if y > 1:
      control = 1
    else:
      messagebox.showinfo("Wrong input", "Please define a proper number of grid points: ")
  control = 0
  while control == 0:
    Re = float(input("Define Reynolds number [>0]"))
    if Re > 0:
      control = 1
    else:
      messagebox.showinfo("Wrong input", "Please define a proper number : ")
  control = 0
  while control == 0:
    E = float(input("Define desired E coefficient: "))
    if 4000 > E > 1 :
      control = 1
    else:
      messagebox.showinfo("Wrong input", "Please define a proper error threshold: [<1]")
  control = 0
  while control == 0:
    error = float(input("Define desired error threshold: [<1]"))
    if 1 > error > 0:
      control = 1
    else:
      messagebox.showinfo("Wrong input", "Please define a proper error threshold: [<1]")
  control = 0
  while control == 0:
    relax = input("Do u wish to add relaxation method?[y, n]")
    if relax == 'y':
      tropos = 1
      control = 1
      control2 = 0
      while control2 == 0:
        omega = float(input("Define desired omega factor: [0 < ω < 2]"))
        if 2 > omega > 0:
          control2 = 1
        else:
          messagebox.showinfo("Wrong input", "Please define a proper value: [0 < ω < 2]")
    elif relax == 'n':
```

```
196.            tropos = 0
197.            control = 1
198.        else:
199.            messagebox.showinfo("Wrong input", "Please give a proper answer: [y, n]")
200.    peplegmeni(y, Re, E, error, omega, tropos)
```