Αεροελαστικότητα & Αεροακουστική
9^{o} εξάμηνο

Προεραιτικό θέμα Αεροελαστικότητας

Χρονική απόκριση ταλάντωσης αεροτομής σε συνθήκες μόνιμης και μη-μόνιμης αεροδυναμικής

Γεώργιος Νιδριώτης - mc18054

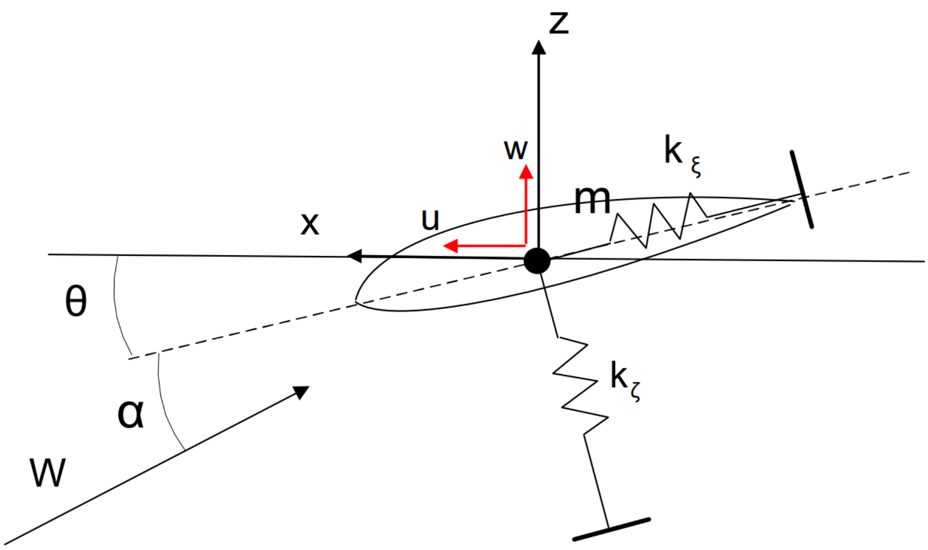
5 Μαρτίου 2024

Περιεχόμενα

Κατάλογος σχημάτων

Εισαγωγή

Η παρούσα εργασία αφορά τον προσδιορισμό της αεροελαστικής συμπεριφοράς διδιάστατης αεροτομής. Συγκεκριμένα, θα ασχοληθούμε με τον προσδιορισμό της αεροδυναμικής απόσβεσης και τη χρονική απόκριση της αεροτομής σε ελεύθερο πεδίο ροής. Η αεροδυναμική μοντελοποίηση της αεροτομής θα γίνει με δύο διαφορετικές μεθόδους, θεωρώντας μόνιμη και αποκατεστημένη ροή, και χρησιμοποιώντας μη-μόνιμο μοντέλο που λαμβάνει υπόψη και μεταβατικά φαινόμενα, ωστόσο περεταίρω εξήγηση δίνεται στις παρακάτω παραγράφους. Αναφορικά με την ελαστική στήριξη της αεροτομής, θεωρούμε τμήμα (μοναδιαίου μήκους) μιας πτέρυγας, και η ελαστική στήριξη αφορά την ανηγμένη δυσκαμψία σε κάμψη της πτέρυγας, στη θέση του τμήματος που μελετάμε. Επιπλέον, θεωρούμε μηδενική δομική απόσβεση (structural damping) και πως οι κύριοι άξονες του μητρώου ακαμψίας της διατομής είναι προσανατολισμένοι παράλληλα, και κάθετα στην χορδή της αεροτομής (σε μηδενική γωνία δεν έχουμε ελαστική σύζευξη). Ένα απλοποιημένο σχήμα του μοντέλου φαίνεται στο σχήμα 1.



Σχήμα 1: Σχηματική αναπαράσταση μοντέλου αεροτομής

1 Μοντελοποίηση - ορισμός αεροελαστικών εξισώσεων

Με το μοντέλο που περιγράφηκε παραπάνω, θα εξάγουμε τις αεροελαστικές εξισώσεις που περιγράφουν την κίνηση της αεροτομής. Οι εξισώσεις διατυπώνονται για τη γενική περίπτωση που έχουμε ελαστική σύζευξη, που ισχύει όταν έχουμε μη-μηδενική γωνία θ. Θεωρώντας δεδομένα την ελαστικότητα (σταθερές ελατηρίων), τη γωνία της αεροτομής, και τη γωνία προσβολής του ανέμου, εφαρμόζωντας την ενεργειακή μέθοδο Lagrange, οι κινητική και η δυναμική ενέργεια του συστήματος είναι:

$$\begin{aligned} T &= \frac{1}{2}m(\dot{u}^2+\dot{w}^2) \\ U &= \frac{1}{2}k_{xi}(u\cdot cos\theta - w\cdot sin\theta)^2 + \frac{1}{2}k_\zeta(u\cdot sin\theta + w\cdot cos\theta)^2 \end{aligned} \tag{1.1}$$

Και με βάση την μέθοδο Lagrange, η εξίσωση ισορροπίας για κάθε Β.Ε είναι

$$\frac{d}{dt}\Big(\frac{\partial T-U}{\partial \dot{q}_i}\Big) - \frac{\partial T-U}{\partial q_i} = Q_i \tag{1.2}$$

Και τελικά οι δύο εξισώσεις καταλήγουν στην παρακάτω μορφή.

$$\begin{aligned} m\ddot{u} + \big(cos^2\theta \cdot k_\xi + sin^2\theta \cdot k_\zeta\big) \cdot u + sin\theta cos\theta \cdot (k_\zeta - k_\xi) \cdot w = F_x \\ m\ddot{w} + sin\theta cos\theta(k_\zeta - k_\xi) \cdot u + \big(sin^2\theta \cdot k_\xi + cos^2\theta \cdot k_\zeta\big) \cdot w = F_z \end{aligned} \tag{1.3}$$

Ή μητρωϊκά,

$$\underbrace{\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}}_{\mathbf{M}} \begin{Bmatrix} \ddot{u} \\ \ddot{w} \end{Bmatrix} + \underbrace{\begin{bmatrix} cos^2\theta \cdot k_\xi + sin^2\theta \cdot k_\zeta & sin\theta cos\theta \cdot (k_\zeta - k_\xi) \\ sin\theta cos\theta \cdot (k_\zeta - k_\xi) & sin^2\theta \cdot k_\xi + cos^2\theta \cdot k_\zeta \end{bmatrix}}_{\mathbf{K}} \begin{Bmatrix} u \\ w \end{Bmatrix} = \begin{Bmatrix} F_x \\ F_z \end{Bmatrix} \tag{1.4}$$

Με βάση την εξίσωση 1.4, σημειώνονται τα εξής. Το μητρώο K αφορά το μετασχηματισμένο μητρώο δυσκαμψίας της αεροτομής απο το τοπικό της σύστημα στο γενικό. Επίσης, οι εξισώσεις 1.3, 1.4 βρίσκονται στην πρωτογενή τους μορφή. Δηλαδή, δεν μπορούμε να ποσοτικοποιήσουμε την επίδραση της αεροδυναμικής στην απόσβεση απο την εξίσωση 1.4.

Αν γραμμικοποιήσουμε την εξίσωση 1.4 γύρω απο μια θέση (έστω τη θέση ισορροπίας), τότε παίρνει την παρακάτω μορφή (εξ. 1.5).

$$[\mathbf{M}]\delta\ddot{\mathbf{u}} + \underbrace{\begin{bmatrix} -\dfrac{\partial F_x}{\partial \dot{u}} & -\dfrac{\partial F_x}{\partial \dot{w}} \\ -\dfrac{\partial F_z}{\partial \dot{u}} & -\dfrac{\partial F_z}{\partial \dot{w}} \end{bmatrix}}_{\mathbf{C}} \delta\dot{\mathbf{u}} + [\mathbf{K}]\delta\mathbf{u} = \mathbf{F}_0 - [\mathbf{M}]\ddot{\mathbf{u}}_\mathbf{0} - [\mathbf{K}]\mathbf{u_0} \tag{1.5}$$

Η μορφή της εξίσωσης 1.5 είναι πολύ χρήσιμη στην ιδιοανυσματική ανάλυση, απο όπου μπορούμε να εξάγουμε πληροφορίες για την ευστάθεια της αεροτομής μέσω του λόγου απόσβεσης. Φυσικά, η εξίσωση 1.5 μας παρέχει πληροφορία για την ευστάθεια στο σημείο γραμμικοποίησης, δηλαδή, αν γραμμικοποιήσουμε σε σημείο κοντά αλλά πριν την αντιστροφή της κλίσης του διαγράμματος $Cl-\alpha$ (βλ. σχήμα 2), η αεροτομή θα φανεί πιθανότατα ευσταθής, παρ'όλο που κατά την ταλάντωσή της μπορεί να βρεθεί σε σημείο όπου έχουμε αρνητική απόσβεση, και σε αυτό θα αναφερθούμε σε επόμενες παραγράφους.

Οπότε, πλέον, με αφετηρία την εξίσωση 1.3 ή 1.4, μπορούμε να καθορίσουμε τις αεροελαστικές εξισώσεις για μόνιμη και μη-μόνιμη ροή.

1.1 Αεροελαστικές εξισώσεις μόνιμης ροής

Σε αυτή την περίπτωση, θα υπολογίσουμε την άνωση και την αντίσταση χρησιμοποιώντας δεδομένα για αεροτομή που προκύπτουν απο μόνιμη πλήρως ανεπτυγμένη ροή. Για την υλοποίηση της εργασίας, χρησιμοποιήσαμε αεροτομή NACA 2412, και προσεγγίσαμε τις καμπύλες $Cl-\alpha, Cd-\alpha$ χρησιμοποιώντας προσομοίωση με συνεκτική ροή στο XFOIL. Στο λογισμικό που αναπτύχθηκε, έγινε προεκβολή για τιμές εκτός του εύρους της προσομοίωσης, με ανάπτυγμα Taylor χρησιμοποιώντας όρους δεύτερης τάξης, και θέσαμε όριο στις $\pm 50^o$ για να αποφύγουμε την αύξηση του σφάλματος για μεγάλες τιμές της γωνίας προσβολής. Για γωνίες μεταξύ αποτελεσμάτων των προσομοιώσεων, οι τιμές των Cl, Cd λαμβάνονται με γραμμική παρεμβολή. Τέλος, οι τιμές της κλίσης, λαμβάνονται με κεντρικές πεπερασμένες διαφορές. Οι καμπύλες Cl, Cd που προκύπτουν φαίνονται στα διαγράμματα 2, 3.

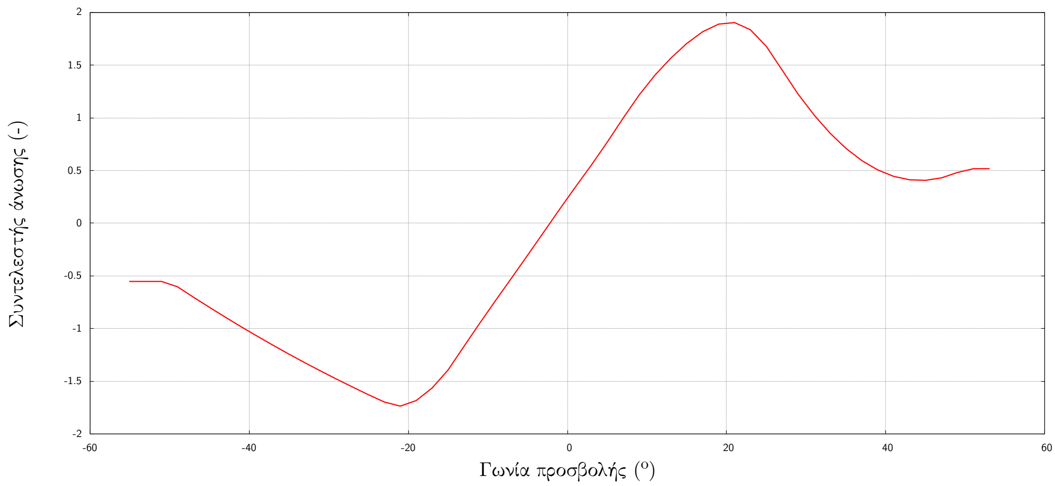
Με δεδομένες καμπύλες των αεροδυναμικών συντελεστών, μπορούμε να υπολογίσουμε τα αεροδυναμικά φορτία. Αρχικά, πρέπει να ορίσουμε τη φαινομενική ταχύτητα της ροής και τη φαινομενική γωνία προσβολής και η έκφρασή τους λαμβάνωντας υπόψη την κίνηση της αεροτομής δίνεται παρακάτω.

$$W_{eff} = \sqrt{\big(-Wcos(\theta+\alpha) - \dot{u}\big)^2 + \big(Wsin(\theta+\alpha) - \dot{w}\big)^2} \tag{1.6}$$

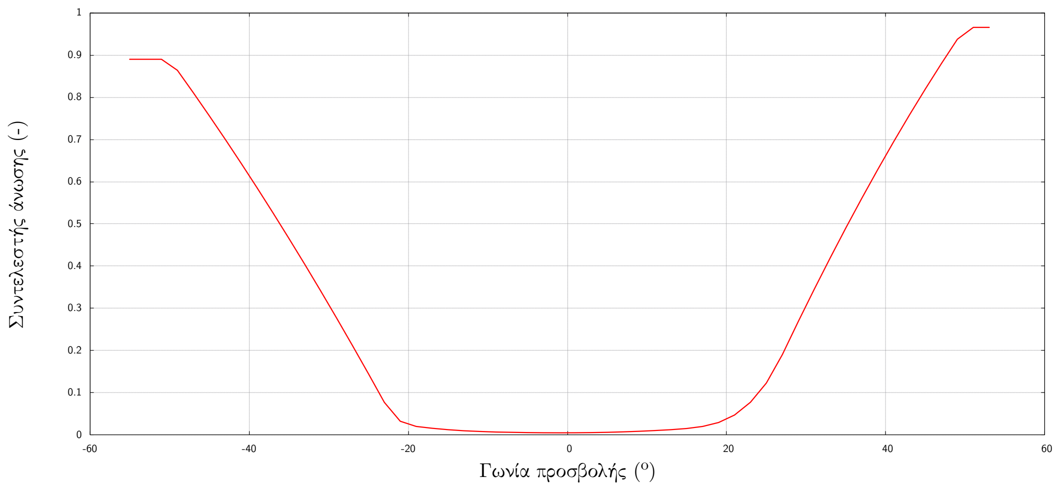
$$\alpha_{eff} = -atan\Big(\frac{Wsin(\alpha+\theta) - \dot{w}}{-Wcos(\alpha+\theta) - \dot{u}}\Big) - \theta \tag{1.7}$$

Και επομένως, κατά τα γνωστά, η δύναμη αντίστασης και άνωσης δίνονται απο τη σχέση 1.8, και μετασχηματισμένες στο γενικό σύστημα απο τη σχέση 1.9.

$$\begin{aligned} L = \frac{1}{2}\rho \cdot c \cdot C_L(\alpha_{eff}) \cdot W_{eff}^2 \\ D = \frac{1}{2}\rho \cdot c \cdot C_D(\alpha_{eff}) \cdot W_{eff}^2 \end{aligned} \tag{1.8}$$



Σχήμα 2: Διάγραμμα συντελεστή άνωσης



Σχήμα 3: Διάγραμμα συντελεστή αντίστασης

$$\begin{aligned} F_x &= L \cdot sin(\alpha_{eff} + \theta) - D \cdot cos(\alpha_{eff} + \theta) \\ F_z &= L \cdot cos(\alpha_{eff} + \theta) + D \cdot sin(\alpha_{eff} + \theta) \end{aligned} \tag{1.9}$$

Επομένως, πλεόν, παραγωγίζωντας τις σχέσεις 1.9 μπορούμε να λάβουμε το μητρώο της αεροδυναμικής απόσβεσης, όπως προκύπτει απο την εξίσωση 1.5.

$$\begin{aligned} \frac{\partial F_x}{\partial \dot{u}} = & \frac{\partial L}{\partial \dot{u}} \cdot sin(\alpha_{eff} + \theta) + L \cdot cos(\alpha_{eff} + \theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \\ & - \frac{\partial D}{\partial \dot{u}} \cdot cos(\alpha_{eff} + \theta) + D \cdot sin(\alpha_{eff} + \theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \end{aligned} \tag{1.10}$$

$$\begin{aligned}\frac{\partial F_x}{\partial \dot{w}} &= \frac{\partial L}{\partial \dot{w}} \cdot sin(\alpha_{eff}+\theta) + L \cdot cos(\alpha_{eff}+\theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}} \\ &- \frac{\partial D}{\partial \dot{w}} \cdot cos(\alpha_{eff}+\theta) + D \cdot sin(\alpha_{eff}+\theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}}\end{aligned} \tag{1.11}$$

Και αντίστοιχα για την F_z.

$$\begin{aligned}\frac{\partial F_z}{\partial \dot{u}} &= \frac{\partial L}{\partial \dot{u}} \cdot cos(\alpha_{eff}+\theta) - L \cdot sin(\alpha_{eff}+\theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \\ &+ \frac{\partial D}{\partial \dot{u}} \cdot sin(\alpha_{eff}+\theta) + D \cdot cos(\alpha_{eff}+\theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}}\end{aligned} \tag{1.12}$$

$$\begin{aligned}\frac{\partial F_z}{\partial \dot{w}} &= \frac{\partial L}{\partial \dot{w}} \cdot cos(\alpha_{eff}+\theta) - L \cdot sin(\alpha_{eff}+\theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}} \\ &+ \frac{\partial D}{\partial \dot{w}} \cdot sin(\alpha_{eff}+\theta) + D \cdot cos(\alpha_{eff}+\theta) \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}}\end{aligned} \tag{1.13}$$

Για τον υπολογισμό των παραγώγων απο τις εξισώσεις (1.10)-(1.13), χρειαζόμαστε τις εκφράσεις των $\frac{\partial L}{\partial \dot{\mathbf{u}}}, \frac{\partial D}{\partial \dot{\mathbf{u}}}, \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}}$. Θεωρώντας πως $W_{eff} \approx W$, οι παραπάνω παράγωγοι είναι.

$$\begin{aligned}\frac{\partial L}{\partial \dot{u}} &= \frac{\partial C_L}{\partial \dot{u}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2 = \underline{\frac{\partial C_L}{\partial \alpha_{eff}} \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2} \\ \frac{\partial L}{\partial \dot{w}} &= \frac{\partial C_L}{\partial \dot{w}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2 = \underline{\frac{\partial C_L}{\partial \alpha_{eff}} \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2}\end{aligned} \tag{1.14}$$

$$\begin{aligned}\frac{\partial D}{\partial \dot{u}} &= \frac{\partial C_D}{\partial \dot{u}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2 = \underline{\frac{\partial C_D}{\partial \alpha_{eff}} \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2} \\ \frac{\partial D}{\partial \dot{w}} &= \frac{\partial C_D}{\partial \dot{w}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2 = \underline{\frac{\partial C_D}{\partial \alpha_{eff}} \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}} \cdot \frac{\rho}{2} \cdot c \cdot W_{eff}^2}\end{aligned} \tag{1.15}$$

Και απο την εξίσωση (1.7) η παράγωγοι $\frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}}$ είναι.

$$\frac{\partial \alpha_{eff}}{\partial \dot{u}} = \frac{-(W sin(\alpha+\theta) - \dot{w})}{W^2 + \dot{u}^2 + \dot{w}^2 + 2 \cdot W \cdot \dot{u} cos(\alpha+\theta) - 2 \cdot W \cdot \dot{w} \cdot sin(\alpha+\theta)} \tag{1.16}$$

$$\frac{\partial \alpha_{eff}}{\partial \dot{w}} = \frac{-(W \cdot cos(\alpha+\theta) + \dot{u})}{W^2 + \dot{u}^2 + \dot{w}^2 + 2 \cdot W \cdot \dot{u} cos(\alpha+\theta) - 2 \cdot W \cdot \dot{w} \cdot sin(\alpha+\theta)} \tag{1.17}$$

Αντικαθιστώντας τις εξισώσεις (1.14)-(1.17) στις εξισώσεις (1.10)-(1.13), προκύπτουν οι τελικές εκφράσεις του μητρώου αεροδυναμικής απόσβεσης.

$$\begin{aligned}\frac{\partial F_x}{\partial \dot{u}} = \frac{\rho}{2} \cdot W^2 \cdot c \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \cdot \Bigg[& sin(\alpha_{eff}+\theta)\Big(\frac{\partial C_L}{\partial \alpha_{eff}} + C_D\Big) + \\ & + cos(\alpha_{eff}+\theta)\Big(C_L - \frac{\partial C_D}{\partial \alpha_{eff}}\Big)\Bigg]\end{aligned} \tag{1.18}$$

$$\frac{\partial F_x}{\partial \dot{w}} = \frac{\rho}{2} \cdot W^2 \cdot c \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}} \cdot \Bigg[sin(\alpha_{eff} + \theta)\Big(\frac{\partial C_L}{\partial \alpha_{eff}} + C_D\Big) + \\ + cos(\alpha_{eff} + \theta)\Big(C_L - \frac{\partial C_D}{\partial \alpha_{eff}}\Big)\Bigg] \tag{1.19}$$

$$\frac{\partial F_z}{\partial \dot{u}} = \frac{\rho}{2} \cdot W^2 \cdot c \cdot \frac{\partial \alpha_{eff}}{\partial \dot{u}} \cdot \Bigg[sin(\alpha_{eff} + \theta)\Big(\frac{\partial C_D}{\partial \alpha_{eff}} + C_L\Big) + \\ + cos(\alpha_{eff} + \theta)\Big(C_D - \frac{\partial C_L}{\partial \alpha_{eff}}\Big)\Bigg] \tag{1.20}$$

$$\frac{\partial F_z}{\partial \dot{w}} = \frac{\rho}{2} \cdot W^2 \cdot c \cdot \frac{\partial \alpha_{eff}}{\partial \dot{w}} \cdot \Bigg[sin(\alpha_{eff} + \theta)\Big(\frac{\partial C_D}{\partial \alpha_{eff}} + C_L\Big) + \\ + cos(\alpha_{eff} + \theta)\Big(C_D - \frac{\partial C_L}{\partial \alpha_{eff}}\Big)\Bigg] \tag{1.21}$$

Και τελικά

$$[\mathbf{C}] = \begin{bmatrix} -\frac{\partial F_x}{\partial \dot{u}} & -\frac{\partial F_x}{\partial \dot{w}} \\ -\frac{\partial F_z}{\partial \dot{u}} & -\frac{\partial F_z}{\partial \dot{w}} \end{bmatrix} \tag{1.22}$$

1.2 Αεροελαστικές εξισώσεις μη-μόνιμης ροής

Για τη διατύπωση των αεροελαστικών εξισώσεων μη-μόνιμης ροής, χρησιμοποιούμε τη διατύπωση των αεροδυναμικών φορτίων που παρέχουν οι εξισώσεις του Theodorsen για προσκολλημένη ροή. Οι εξισώσεις του Theodorsen, προσεγγίζουν την αεροτομή ως μια διδιάστατη επιφάνεια (επίπεδη πλάκα χωρίς πάχος), και θεωρούν πως η αεροδυναμική συμπεριφορά της αεροτομής κυριαρχείται απο την κίνησή της κάθετα στην χορδή της. Στην τελική διατύπωση τους, οι εξισώσεις του Theodorsen λαμβάνουν υπόψη την επίδραση του ομόρρου στο οριακό στρώμα της πτέρυγας και επιπλέον συνυπολογίζουν την προστιθέμενη μάζα που προκύπτει απο το ρευστό λόγω της επιτάχυνσης της πτέρυγας (κάθετα στην χορδή). Οι εξισώσεις του Theodorsen στην αρχική τους μορφή, θεωρούν και στροφικό βαθμό ελευθερίας της πτέρυγας, ωστόσο ακολουθώντας τις αρχικές θεωρήσεις μας, όλοι οι αντίστοιχοι όροι έχουν μηδενισθεί. Η έκφραση του συντελεστή άνωσης απο τις εξισώσεις του Theodorsen παρουσιάζεται στην εξίσωση (1.23).

$$C_L = 2\pi\Big(\alpha_E(t) - \alpha_0\Big) - \underbrace{\frac{\pi \cdot c \cdot \ddot{h}}{2 \cdot W_{eff}^2}}_{\text{Όρος προστιθέμενης μάζας}} \tag{1.23}$$

Όπου $\ddot{\mathbf{h}} = \ddot{w}cos\theta + \ddot{u}sin\theta$ η επιτάχυνση της πτέρυγας κάθετα στη χορδή. Σημειώνεται πως για αύξηση της ακρίβειας, ο συντελεστής 2π μπορεί να αντικατασταθεί απο την κλίση της καμπύλης C_L--α στη γραμμική περιοχή, δηλαδή $\left.\frac{dC_L}{d\alpha}\right|_{\text{γραμμ}}$

Επιπλέον, λόγω της φαινόμενης γωνίας προσβολής, επάγεται αντίσταση στο σύστημα παράλληλο με τη χορδή, και ο συντελεστής αντίστασης είναι

$$C_D = C_{D,\text{μον}}(\alpha_E) + C_L \cdot (\alpha_{eff} - \alpha_E) \tag{1.24}$$

Η φαινόμενη γωνία προσβολής (α_E) είναι ο όρος που λαμβάνει υπόψη την επίδραση του ομόρρου στο οριακό στρώμα της αεροτομής και συχνά καλείται όρος ανακυκλοφορίας. Αναπτύσσωντας την ποσότητα α_E έχουμε:

$$\alpha_E = C(k)\alpha_{eff} \tag{1.25}$$

Με

$$\alpha_{eff} = \alpha - \theta - \frac{\dot{h}}{W_{eff}^2} \tag{1.26}$$

Όπου, $C(k)$ αποκαλούμε τη συνάρτηση Theodorsen, που συσχετίζει την ταλάντωση του πεδίου ροής στον ομόρρου με την πίεση στην επιφάνεια της αεροτομής, δηλαδή ποσοτικοποιεί την επίδραση της ταλάντωσης στον ομόρρου στον συντελεστή άνωσης.

Καθώς ο υπολογισμός της συνάρτησης Theodorsen είναι αρκετά επίπονος, θα υπολογίσουμε την φαινόμενη γωνία προσβολής α_E απο την εξίσωση (1.27), όπου y_1, y_2 δύο επιπλέον αεροδυναμικοί βαθμοί ελευθερίας, που προκύπτουν απο τη λύση του συστήματος Σ.Δ.Ε 1.28.

$$\alpha_E = \alpha_{eff} \cdot (1 - A_1 - A_2) + y_1 + y_2 \tag{1.27}$$

$$\begin{aligned} \dot{y_1} + b_1 \frac{2W_{eff}}{c} y_1 - b_1 A_1 \frac{2W_{eff}}{c} \alpha_{eff} = 0 \\ \dot{y_2} + b_2 \frac{2W_{eff}}{c} y_2 - b_2 A_2 \frac{2W_{eff}}{c} \alpha_{eff} = 0 \end{aligned} \tag{1.28}$$

Με $A_1 = 0.165$, $A_2 = 0.335$, $b_1 = 0.0455$, $b_2 = 0.3000$.

Ως οριακές συνθήκες των εξισώσεων 1.28 ορίζουμε πως για $t = 0$ έχουμε $\alpha_E = \alpha_{eff}$ οπότε προκύπτει $y_1(0) = A_1\alpha_{eff}$ και $y_2(0) = A_2\alpha_{eff}$. Οι εξισώσεις 1.28 διακριτοποιούνται με πεπερασμένες διαφορές και υπολογίζουμε τις τιμές των y_1, y_2 πριν τον υπολογισμό των εξωτερικών φορτίων από την εξίσωση (1.9).

Για να υπολογίσουμε πλέον, τις παραγώγους των αεροδυναμικών φορτίων και συνεπώς το μητρώο αεροδυναμικής απόσβεσης διαφορίζουμε τις Δ.Ε 1.28.

$$\frac{\partial}{\partial \dot{u}}\left(\frac{dy_1}{dt}\right) + b_1 \frac{2W_{eff}}{c} \frac{\partial y_1}{\partial \dot{u}} - b_1 A_1 \frac{2W_{eff}}{c} \frac{\partial \alpha_{eff}}{\partial \dot{u}} = 0 \tag{1.29}$$

Θεωρώντας τον όρο δεύτερης τάξης $\frac{\partial}{\partial \dot{u}}\left(\frac{dy_1}{dt}\right) \approx 0$ προκύπτει:

$$\frac{\partial y_1}{\partial \dot{\mathbf{u}}} = A_1 \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}} \tag{1.30}$$

$$\frac{\partial y_2}{\partial \dot{\mathbf{u}}} = A_2 \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}} \tag{1.31}$$

Επιπλέον,

$$\frac{\partial \alpha_E}{\partial \dot{\mathbf{u}}} = (1 - A_1 - A_2) \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}} + \frac{\partial y_1}{\partial \dot{\mathbf{u}}} + \frac{\partial y_2}{\partial \dot{\mathbf{u}}} \tag{1.32}$$

Και αντικαθιστώντας τις εξισώσεις (1.30)-(1.31), προκύπτει:

$$\frac{\partial \alpha_E}{\partial \dot{\mathbf{u}}} = \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}} \tag{1.33}$$

Ενώ απο την εξίσωση (1.26) είναι:

$$\begin{aligned} \frac{\partial \alpha_{eff}}{\partial \dot{u}} &= -sin(\theta) \\ \frac{\partial \alpha_{eff}}{\partial \dot{w}} &= -cos(\theta) \end{aligned} \tag{1.34}$$

Επομένως, τελικά έχουμε

$$\frac{\partial C_L}{\partial \dot{\mathbf{u}}} = \left.\frac{dC_L}{d\alpha}\right|_{\text{γραμμ}} \cdot \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}} \tag{1.35}$$

Και για τον συντελεστή αντίστασης

$$\frac{\partial C_D}{\partial \dot{\mathbf{u}}} = \frac{\partial C_D}{\partial \alpha} \cdot \frac{\partial \alpha_{eff}}{\partial \dot{\mathbf{u}}} + \frac{\partial C_L}{\partial \dot{\mathbf{u}}} \cdot (\alpha_{eff} - \alpha_E) \tag{1.36}$$

Και οι τελικές τιμές των παραγώγων των αεροδυναμικών φορτίων υπολογίζονται απο τις εξισώσεις (1.10)-(1.15). Σημειώνεται ωστόσο, πως η θεώρηση του όρου $\frac{\partial}{\partial \ddot{u}}\left(\frac{dy_1}{dt}\right) \approx 0$ αγνοεί την επιρροή του ομόρρου στην αεροδυναμική απόσβεση της αεροτομής και αποτελεί μια παραδοχή για απλοποίηση των υπολογισμών και την αποφυγή της επίλυσης μιας ακόμη Σ.Δ.Ε. Ωστόσο, όπως θα δούμε και σε επόμενη παράγραφο, αν δεν λύσουμε την γραμμικοποιημένη εξίσωση για τον προσδιορισμό της χρονοσειράς των αποκρίσεων, αλλά χρησιμοποιήσουμε την αρχική έκφραση, αυτή η απλοποίηση δεν αποτελεί αίτιο προβληματισμού. Ωστόσο, αν θέλαμε να συγκρίνουμε τις τιμές της απόσβεσης που δίνουν οι εκφράσεις των μόνιμων και μη-μόνιμων μοντέλων, θα ήταν αναγκαίο να συμπεριλάβουμε και τον όρο δεύτερης τάξης.

2 Ιδιοανυσματική ανάλυση συστήματος

2.1 Υπολογισμός ιδιοτιμών συστήματος

Για την αναγνώριση των ιδιοσυχνοτήτων και της απόσβεσης του συστήματος πραγματοποιούμε ιδιοανυσματική ανάλυση.
Αρχικά, μετασχηματίζουμε τις αεροελαστικές εξισώσεις σε μορφή κατάστασης-χώρου (state space transformation).

$$\begin{aligned} \mathbf{y_1} &= \dot{\mathbf{x}} \\ \mathbf{y_2} &= \mathbf{x} \end{aligned} \tag{2.1}$$

Επομένως το σύστημα γίνεται

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \dot{\begin{Bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{Bmatrix}} = \begin{bmatrix} -\mathbf{C} & -\mathbf{K} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \cdot \begin{Bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{Bmatrix} + \begin{Bmatrix} \mathbf{Q} \\ \mathbf{0} \end{Bmatrix} \tag{2.2}$$

Και εναλλακτικά

$$\dot{\begin{Bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{Bmatrix}} = \underbrace{\begin{bmatrix} -\mathbf{M}^{-1}\mathbf{C} & -\mathbf{M}^{-1}\mathbf{K} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}}_{A} \cdot \begin{Bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{Bmatrix} + \underbrace{\begin{bmatrix} \mathbf{M}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{Bmatrix} \mathbf{Q} \\ \mathbf{0} \end{Bmatrix}}_{B} \tag{2.3}$$

Αναλύοντας το ομογενές σύστημα, οι ιδιοσυχνότητες και η απόσβεση δίνονται απο τις ιδιοτιμές του μητρώου **A**. Για τους δυο βαθμούς ελευθερίας του φυσικού μας προβλήματος, το μητρώο A έχει διαστάσεις $4x4$ και συνεπώς δίνει τέσσερις ιδιοτιμές -- δύο ζεύγη μιγαδικών συζυγών ιδιοτιμών απο την λύση της εξίσωση (2.4).

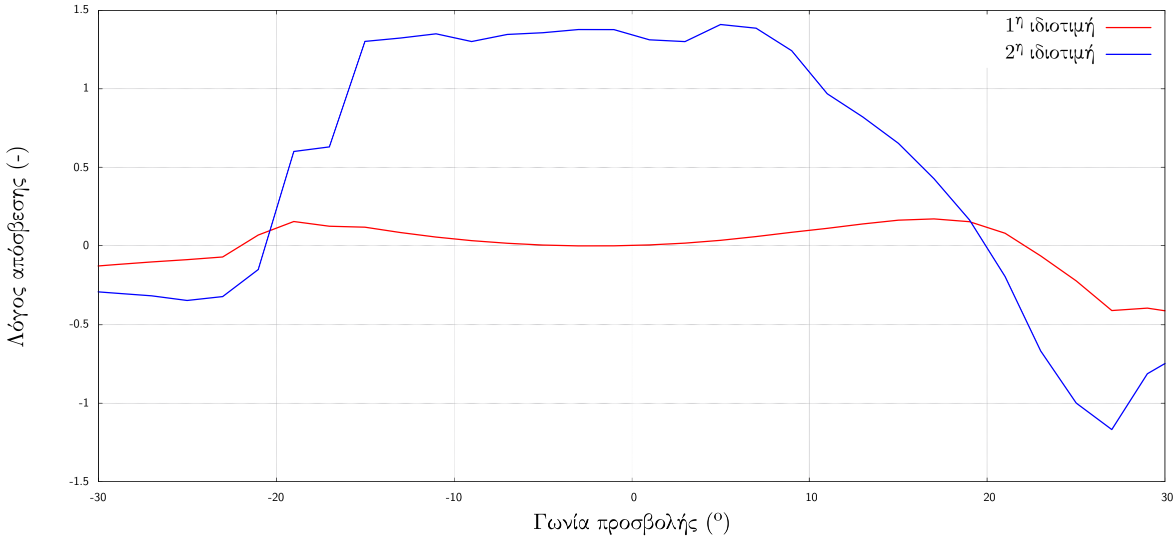
$$det(\mathbf{A} - \lambda_i \mathbf{I}) = 0 \tag{2.4}$$

Σημειώνεται πως στην πράξη, για μητρώα μεγαλύτερα του $2x2$ οι ιδιοτιμές δεν βρίσκονται λύνοντας την εξίσωση (2.4) αλλά βρίσκονται αριθμητικά απο το μητρώο **A**.
Κάθε ζεύγος συζυγών ιδιοτιμών πλεον έχει την παρακάτω μορφή (εξίσωση (2.5)).

$$\lambda_i, \lambda_i^c = -\zeta \pm i \underbrace{\omega_n \sqrt{1 - \zeta^2}}_{\omega_d} \tag{2.5}$$

Επομένως, το πραγματικό μέρος της ιδιοτιμής μας πληροφορεί για την απόσβεση του συστήματος για συχνότητα ταλάντωσης ω_d και το φανταστικό μέρος μα δίνει την ιδιοσυχνότητα του συστήματος για κάθε ιδιοτιμή.
Επιπλέον, απο την εξίσωση (2.6) για κάθε ιδιοτιμή μπορούμε να βρούμε την αντίστοιχη ιδιομορφή, που μας πληροφορεί για το πλάτος της ταλάντωσης των βαθμών ελευθερίας όταν το σύστημα ταλαντώνεται με την αντίστοιχη ιδιοσυχνότητα. Στη δική μας περίπτωση για κάθε ιδιοτιμή μπορούμε να βρούμε την αντίστοιχη ιδιομορφή, που μας πληροφορεί για τη συσχέτιση των βαθμών ελευθερίας όταν το σύστημα ταλαντώνεται με την αντίστοιχη ιδιοσυχνότητα. Κάθε ιδιομορφή συνήθως περιγράφει την κίνηση του συστήματος όπως κυριαρχείται από την ταλάντωση ενος βαθμού ελευθερίας. Σε προβλήματα περισσότερων βαθμών ελευθερίας, και ειδικά σε προβλήματα συνεχούς μέσου, υψηλότερες ιδιοτιμές περιγράφουν και πεπλεγμένες μορφές.

$$(\mathbf{A} - \lambda_i \mathbf{I})\phi_{\mathbf{i}} = 0 \tag{2.6}$$



Σχήμα 4: Μεταβολή του λόγου απόσβεσης συναρτήσει της γωνίας προσβολής

2.2 Υπολογισμός απόσβεσης συναρτήσει της γωνίας προσβολής

Όπως αναφέραμε και στην προηγούμενη παράγραφο, οι ιδιοτιμές του συστήματος υπολογίζονται απο το μητρώο **A** της εξίσωση (2.3). Προφανώς, το μητρώο της απόσβεσης αφορά τη θέση αναφοράς γύρω απο την οποία γραμμικοποιήσαμε, και επιλέξαμε να χρησιμοποιήσουμε τις εξισώσεις μόνιμης ροής (που δεν εξαρτώνται απο την ταχύτητα ταλάντωσης) γραμμικοποιώντας γύρω απο το σημείο ισορροπίας $(\dot{u}, \dot{w} = 0)$.
Τα διαγράμματα μεταβολής της απόσβεσης για τους δυο βαθμούς ελευθερίας συναρτήσει της γωνίας προσβολής παρατίθενται στο σχήμα 4 ενώ τα δεδομένα που χρησιμοποιήθηκαν για το μοντέλο παρατίθενται στον πίνακα 1.

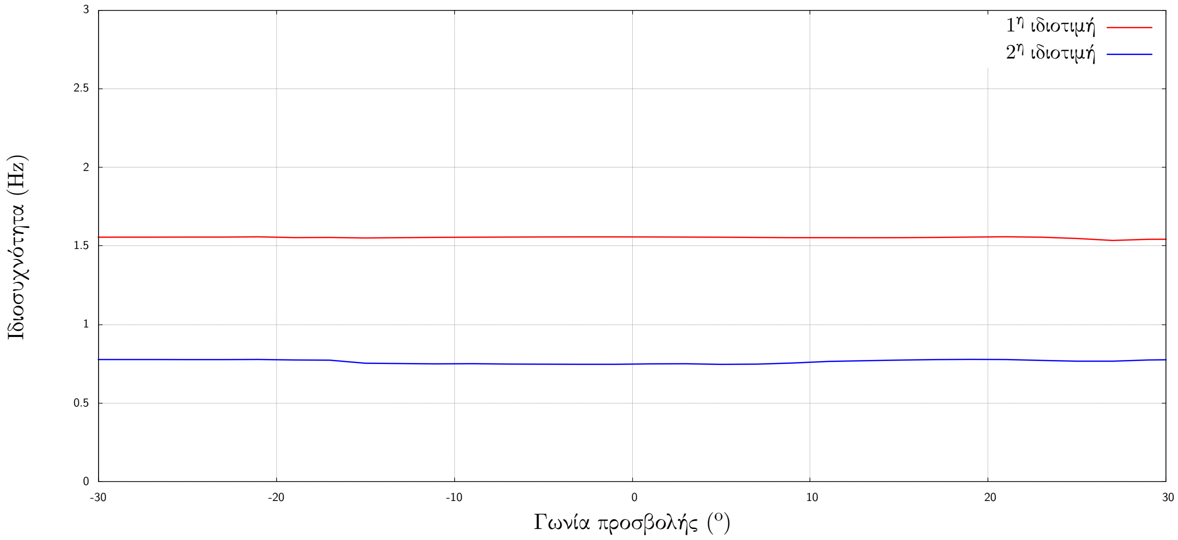
Γραμμική μάζα	**m**	165	kg/m
Ελαστικότητα ∥ χορδής	$\mathbf{k}_\xi$	15791	N/m
Ελαστικότητα ⊥ χορδής	$\mathbf{k}_\zeta$	3948	N/m
Ταχύτητα ελεύθερης ροής	$\mathbf{U_{inf}}$	80	m/s
Γωνία στήριξης	θ	2	^{o}
Γωνία προσβολής	α	4	^{o}
Μήκος χορδής	**c**	1.5	m
Αριθμός Reynolds	**Re**	$8 \cdot 10^6$	--

Πίνακας 1: Δεδομένα μοντέλου

Η απόσβεση του συστήματος γίνεται αρνητική, δηλαδή το σύστημα γίνεται ασταθές, όταν ένας εκ των δύο λόγους απόσβεσης γίνει αρνητικός, διότι αυτό υποδηλώνει πως προσδίδεται ενέργεια στην αεροτομή απο το ρευστό. Στο σχήμα 4, όπως υποδηλώνει και η εξίσωση (2.5) βλέπουμε το πραγματικό μέρος της ιδιοτιμής και φαίνεται να γίνεται αρνητικό για γωνία προσβολής περίπου 20^o και $\approx -21^o$ όταν η ιδιοτιμή που αντιστοιχεί στον Β.Ε πτερύγησης γίνεται αρνητική. Παρατηρούμε επίσης, πως το μεγαλύτερο μέρος της απόσβεσης παρέχεται απο τον Β.Ε πτερύγησης (flapwise) , ενώ για την chordwise κίνηση έχουμε μικρές αλλά θετικές τιμές απόσβεσης. Τέλος, παρατηρούμε πως οι ιδιοσυχνότητες επηρεάζονται ελάχιστα απο τη γωνία προσβολής.

Για τα δεδομένα του προβλήματος μας (πίνακας 1) οι δυο ιδιοσυχνότητες και λόγοι απόσβεσης φαίνονται στον πίνακα 2.

Τα ιδιοδιανύσματα που προκύπτουν απο την ανάλυση οπτικοποιούνται στο σχήμα 6. Ο βρόγχος που σχηματίζεται

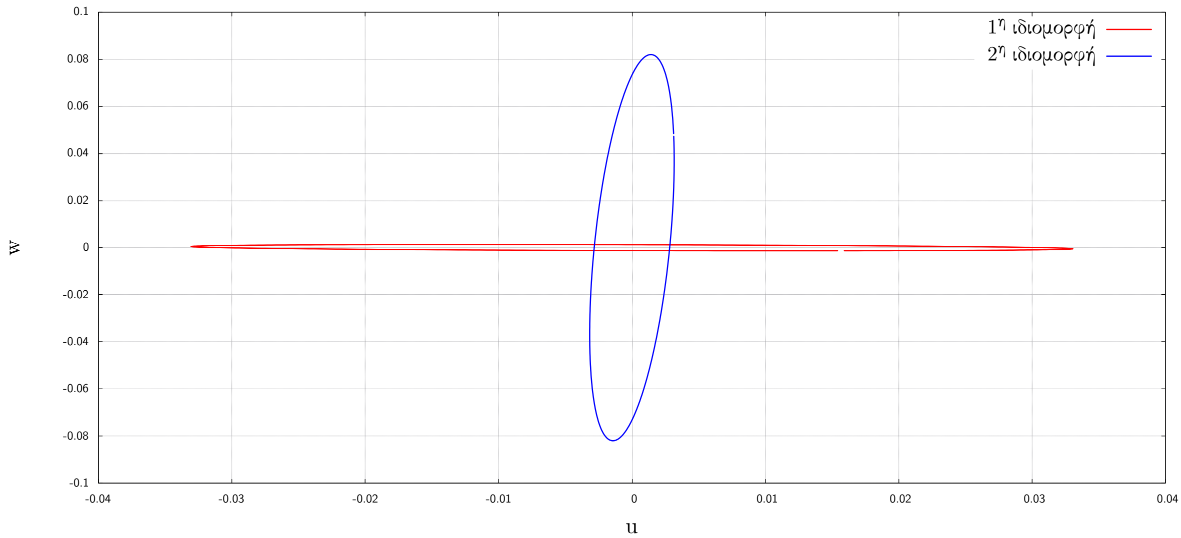


Σχήμα 5: Μεταβολή των ιδιοσυχνοτήτων συναρτήσει της γωνίας προσβολής

	1^{η} ιδιοτιμή	2^{η} ιδιοτιμή
Ιδιοσυχνότητα (Hz)	1.55	0.75
Λόγος απόσβεσης (--)	0.037	1.325

Πίνακας 2: Ιδιοτιμές συστήματος

υποδηλώνει την παρουσία απόσβεσης, και υπεισέρχεται ως διαφορά φάσης μεταξύ των ταλαντώσεων με τις δυο ιδιοσυχνότητες. Όπως είναι φανερό και απο τον πίνακα **??** η πρώτη ιδιομορφή εμφανίζει πολύ μικρότερο βρόγχο που συμφωνεί με τον μικρότερο λόγο απόσβεσης της ιδιοτιμής.



Σχήμα 6: Οπτικοποίηση ιδιοδιανυσμάτων για τις δυο ιδιοτιμές

3 Χρονική ολοκλήρωση των αεροελαστικών εξισώσεων

Η χρονική απόκριση των δυο βαθμών ελευθερίας εκφράζονται ως εξής:

$$\mathbf{x}(t) = \mathbf{x}_{hom} + \mathbf{x}_{part} \tag{3.1}$$

Όπου $\mathbf{x}_{hom}$ η ομογενής λύση του συστήματος και $\mathbf{x}_{part}$ η μερική λύση που ακολουθεί τη μορφή της διέγερσης -- των αεροδυναμικών φορτίων.

Η ομογενής λύση, χρησιμοποιώντας τον ιδιοανυσματικό μετασχηματισμό είναι:

$$\mathbf{x}(t)_{hom} = \sum_{k=1}^{2} \left[\phi_{0k} e^{Re(\lambda_k)} \Big(A_k cos(Im(\lambda_k)t + \theta_k) + B_k sin(Im(\lambda_k)t + \theta_k) \Big) \right] \tag{3.2}$$

Όπου, με k συμβολίζουμε την k-οστή ιδιοτιμή, ϕ_{0k} είναι το μέτρο της μιγαδικής ιδιομορφής και θ_k η φάση της. Δηλαδή, η ομογενής λύση προκύπτει απο υπέρθεση των ταλαντώσεων για όλες τις ιδιοτιμές. Οι συντελεστές A_k, B_k προσδιορίζονται απο τις αρχικές συνθήκες.

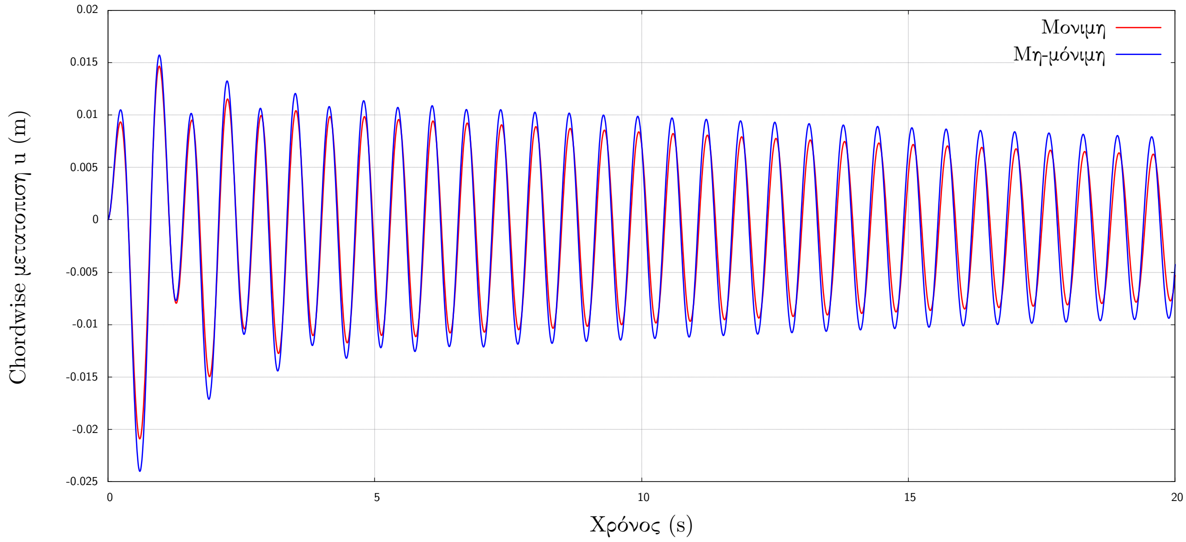
Η αντίστοιχη μερική λύση, θεωρώντας σταθερά αεροδυναμικά φορτία είναι η μόνιμη απόκριση του συστήματος και είναι:

$$\mathbf{x}_{part} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}^{-1} \end{bmatrix} \cdot \begin{Bmatrix} 0 \\ 0 \\ F_x \\ F_z \end{Bmatrix} \tag{3.3}$$

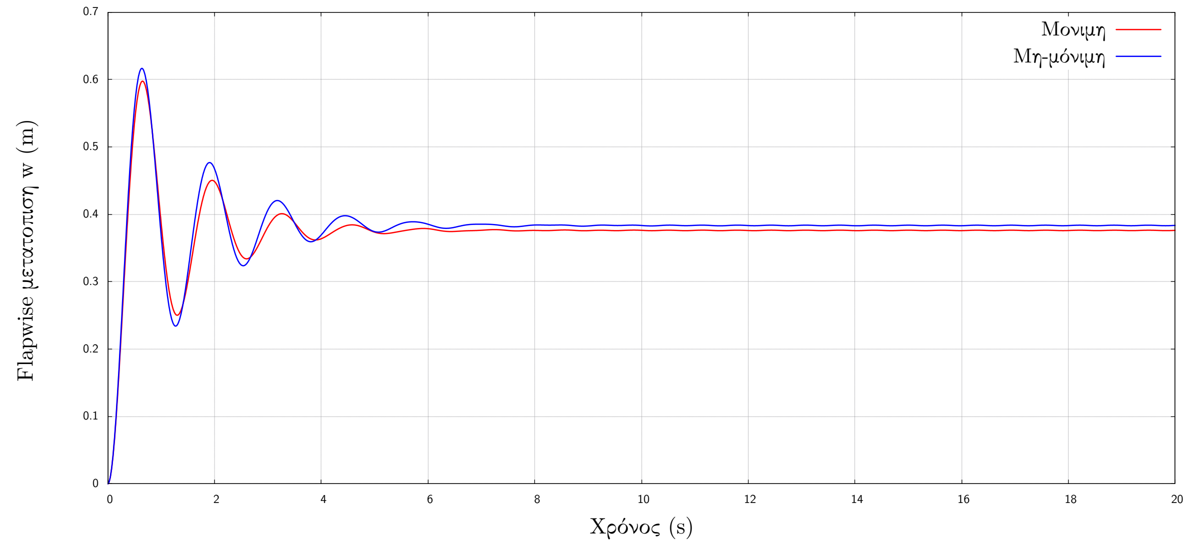
Επομένως, με δεδομένα τα παραπάνω, μπορούμε να προσδιορίσουμε την απόκριση του συστήματος. Για την περίπτωση της μόνιμης ροής, γραμμικοποιήσαμε γύρω απο τη θέση ισορροπίας, και χρησιμοποιήσαμε τις παραπάνω εξισώσεις για τον υπολογισμό της χρονοσειράς της απόκρισης, με απόσβεση και σταθερό αεροδυναμικό φορτίο όπως υπολογίζεται στη θέση αναφοράς.

Για την περίπτωση της μη-μονιμης ροής, όπως αναφέραμε και σε προηγούμενη παράγραφο, επειδή για τον υπολογισμό των αεροδυναμικών φορτίων απαιτείται η επίλυση μιας ακόμη διαφορικής εξίσωσης, σε κάθε χρονικό βήμα, με αρχική συνθήκη την κατάσταση του προηγούμενου βήματος, υπολογίζουμε τα αεροδυναμικά φορτία, και προσδιορίζουμε τη νεα κατάσταση απο τις εξισώσεις (3.1)-(3.3). Εδώ κάνουμε την απλούστευση πως για τη διάρκεια κάθε βήματος, τα αεροδυναμικά φορτία είναι σταθερά και υπολογίζουμε τη μερική λύση απο την εξίσωση (3.3). Επιπλέον, σε αυτή την περίπτωση, στην ομογενή λύση χρησιμοποιούμε μηδενικό μητρώο απόσβεσης, αφού η απόσβεση υπεισέρχεται έμμεσα απο την μεταβολή των αεροδυναμικών φορτίων ανάλογα της κατάστασης του συστήματος.

Ξεκινώντας με μηδενικές μετατοπίσεις, ταχύτητες, και επιταχύνσεις, και χρησιμοποιώντας τα δεδομένα όπως παρατίθενται στον πίνακα 1, η χρονοσειρά της απόκρισης για μόνιμη και μη-μόνιμη ροή, φαίνονται στα σχήματα 7-8.



Σχήμα 7: Χρονοσειρά απόκρισης -- μετατόπιση u



Σχήμα 8: Χρονοσειρά απόκρισης -- μετατόπιση w

Παρατηρούμε πως τα δυο μοντέλα παρέχουν κοντινές αποκρίσεις. Επιπλέον, ειναι προφανές πως κατά την chordwise διεύθυνση έχουμε πολύ μικρότερη απόσβεση, σημειώνοντας πως για χρόνο t=20s η ταλάντωση έχει αποσβεσθεί ελάχιστα. Αντίθετα, όπως υποδεικνύουν και οι τιμές της απόσβεσης που προκύπτουν απο την ιδιανυσματική ανάλυση, κατά την διεύθυνση πτερύγησης έχουμε πολύ εντονότερη απόσβεση. Το γεγονός οτι έχουμε πολύ ασθενή σύζευξη μεταξύ των δυο βαθμών ελευθερίας συντελεί στην ασθενή απόσβεση στη μετατόπιση u. Αν είχαμε εντονότερη σύζευξη, είτε επιλέγοντας μεγαλύτερη γωνία θ ή καλύτερα τροποποιώντας τη διατομή της πτέρυγας ώστε να τροποποιήσουμε τους κύριους άξονες, τότε η ταλάντωση στην chordwise διεύθυνση θα αποσβαινόταν πολύ γρηγορότερα, επάγοντας μετατοπίσεις στην flapwise διεύθυνση όπου έχουμε υψηλή απόσβεση.

Αναφορικά με τη σύγκριση των αποτελεσμάτων των δυο μοντέλων, αρχικά αναφέρεται πως η προσέγγιση της συμπεριφοράς C_L ως γραμμική δίνει ελαφρά διαφορετικές τιμές συγκριτικά με την καμπύλη της μόνιμης ροής, και κατοπτρίζεται σε ελαφρά διαφορετικό σημείο ισορροπίας όταν αποσβαίνεται η ταλάντωση, ωστόσο θα μπορούσε να αντιμετωπισθεί με πιο ακριβή προσδιορισμό της κλίσης της γραμμικής περιοχής του C_L και του σημείου α_0. Επιπλέον, το μοντέλο μη-μόνιμης ροής φαίνεται να είναι πιο συντηρητικό συγκριτικά με τη μόνιμη ροή, αφού παρατηρούμε μεγαλύτερα πλάτη ταλάντωσης, ωστόσο στη συγκεκριμένη περίπτωση μπορεί να οφείλεται στην ελαφρώς μεγαλύτερη τιμή του C_L όπως αναφέραμε παραπάνω.

I Πηγαίος κώδικας

Το λογισμικό που λύνει το μοντέλο υλοποιήθηκε σε γλώσσα C++. Η δομή του κώδικα αποτελείται απο τα εξής. Ενα αρχείο (main.cpp) με την κύρια συνάρτηση του προγράμματος που περιλαμβάνει την κεντρική δομή του αλγορίθμου, και δευτερεύοντα βοηθητικά αρχεία που περιέχουν τους ορισμούς των συναρτήσεων που καλούνται στην κεντρική δομή του αλγορίθμου. Οι παράμετροι του προβλήματος και τα μητρώα μάζας και δυσκαμψίας διαβάζονται απο αντίστοιχα αρχεία εισόδου. Έτσι, παρακάτω παρατίθενται τα αρχεία με την σειρά που αναφέρθηκαν.

I.i Κύρια συνάρτηση αλγορίθμου main.cpp

```cpp
#include "utilities.h"
#include <complex>
#include <fstream>
#include <ostream>


typedef std::complex<double> cmp;

int main(int argc, char *argv[]) {

    double time = std::stod(argv[1]);
    int sz;

    sz = 2;
    // Calculation classes construction
    inputData data(sz);
    inputData alt(sz);

    std::string filename = "matrix.dat";


    // Read inputs
    readMat(sz, data.M, data.K, filename);
    readMat(sz, alt.M, alt.K, filename);

    data.readInput("input.dat");
    alt.readInput("input.dat");

    // Read aerodynamic data
    std::string clpath = "doc/CL.dat";
    std::string cdpath = "doc/CD.dat";
    readAero(clpath, cdpath, data);
    readAero(clpath, cdpath, alt);

    // Initialize solutions
    // Calculates Damping, loads
    // and steady state solution
    data.initialize();
    alt.initialize();
    data.theodor = true; // data class will use Theodorsen equations

    // □□□□□□□□□□ □□□ □□□□□□□□□□□□ □□□□□□□ □□□□□□

        std::ofstream outE("theodorsen.dat");
        outE << "Time (s) " << std::setw(25) << "U" << std::setw(25) << "W" << std::endl;
        outE.close();

        std::ofstream outA("eigenSteady.dat");
        outA << "Time (s) " << std::setw(25) << "U" << std::setw(25) << "W" << std::endl;
        outA.close();

    // Unsteady class calculation initialization
    Theodorsen theodorsen(&data);

    // Main time integration loop
    while (data.t_tot < time)
    {
        std::cout << data.t_tot << "s" << std::setw(25) << data.eigenvalues[0] << std::setw(25) << data.eigenvalues[1] << std::endl;

```

```
        theodorsen.calcNextY(); // Calculate next discrete y1, y2
        theodorsen.calcLoads();
        data.computeEigen();
        data.computeConstants(); // Compute homogenous solution coefficients
        data.nextAnalyticalStep("theodorsen.dat");

        alt.computeConstants();
        alt.nextAnalyticalStep("eigenSteady.dat");
    }


    // Angle of attack vs Damping

    // std::ofstream al("damping.dat");
    // std::ofstream freq("frequency.dat");
    // eigen.alpha = -pi/4;
    // while (eigen.alpha < pi/4) {
    //     eigen.calcDamp();
    //     eigen.computeEigen();
    //
    //     al << eigen.alpha/pi*180 << " " << eigen.eigenvalues[0].real() << " " << eigen.
    eigenvalues[1].real() << std::endl;
    //     freq << eigen.alpha/pi*180 << " " << eigen.eigenvalues[0].imag() << " " << eigen.
    eigenvalues[1].imag() << std::endl;
    //
    //     eigen.alpha += 2*pi/180;
    //
    // }
    // al.close();
    // freq.close();

}
```

I.ii Βοηθητικό αρχείο συναρτήσεων utilities.cpp

```
#include "utilities.h"
#include <Eigen/src/Eigenvalues/GeneralizedEigenSolver.h>
#include <fstream>
#include <istream>


// Reads mass and stiffness matrices
void readMat(int sz, Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> &M, Eigen::Matrix<
    double, Eigen::Dynamic, Eigen::Dynamic> &K, std::string filename)
{
    std::ifstream mat(filename);
    std::string a,b,c,trash;

    mat >> trash;

    for (int i = 0; i < sz; i++) {
        mat >> a >> b ;
        M(i,0) = std::stod(a);
        M(i,1) = std::stod(b);
    }

    mat >> trash;
```

```
    for (int i = 0; i < sz; i++) {
        mat >> a >> b ;
        K(i,0) = std::stod(a);
        K(i,1) = std::stod(b);
    }

}

// Reads input file
void inputData::readInput(std::string filename) {
    std::ifstream inp(filename);

    std::string trash, text;

    getline(inp,trash);
    getline(inp,text);
    U_inf = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    theta = std::stod(text)*pi/180;

    getline(inp,trash);
    getline(inp,text);
    alpha = std::stod(text)*pi/180;

    getline(inp,trash);
    getline(inp,text);
    rho = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    chord = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    u = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    uv = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    ua = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    w = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    wv = std::stod(text);

    getline(inp,trash);
    getline(inp,text);
    wa = std::stod(text);

    // MISPLACED UTILITY

    // Transform stiffness matrix
    Eigen::Matrix<double,2 ,2> R;
    R(0,0) = cos(theta);
    R(0,1) = sin(theta);
    R(1,0) = -sin(theta);
    R(1,1) = cos(theta);

    K = R.transpose()*K*R;
}

// Reads aerodynamic data (curves)
void readAero(std::string filenameCl, std::string filenameCd, inputData &input) {
```

```

    int Lsz;
    int Dsz;
    std::string textCl;
    std::string textCd;

    std::ifstream cl(filenameCl);
    std::ifstream cd(filenameCd);

    // First entry is the count of the data points
    cl >> Lsz;
    cd >> Dsz;

    input.Lift.conservativeResize(Lsz,2);
    input.Drag.conservativeResize(Dsz,2);

    for (int i = 0; i < Lsz; i++)
    {
        cl >> input.Lift(i,0) >> input.Lift(i,1);
    }

    for (int i = 0; i < Dsz; i++)
    {
        cd >> input.Drag(i,0) >> input.Drag(i,1);
    }
}

double inputData::getCL(double a /*Radians*/){
    double alph = a/pi*180;
    double cl;
    int i = 0;
    int sz = Lift.rows()-1;

    if (alph > 50) alph = 50;
    if (alph < -50) alph = -50;

    // Find section for linear interpolation
    while (i < Lift.rows()) {
        if (alph <= Lift(i,0))
        {
            break;
        }
        i++;
    }

    // If a is outside of data bounds
    // 2nd order extrapolation

    double grad0 = (Lift(1,1)-Lift(0,1))/(Lift(1,0)-Lift(0,0));
    double grad0_1 = (Lift(2,1)-Lift(0,1))/(Lift(2,0)-Lift(0,0));

    double gradn = (Lift(sz,1)-Lift(sz-1,1))/(Lift(sz,0)-Lift(sz-1,0));
    double gradn_1 = (Lift(sz,1)-Lift(sz-2,1))/(Lift(sz,0)-Lift(sz-2,0));

    double curv0 = (grad0_1-grad0)/(Lift(1,0)-Lift(0,0));
    double curvn = (gradn-gradn_1)/(Lift(sz,0)-Lift(sz-1,0));

    if (i==0) {
        // Extrapolation using curvature
        cl = Lift(0,1) + grad0*(alph-Lift(0,0)) + pow(alph-Lift(0,0),2)/2*curv0;
    } else if (i == Lift.rows()) {
        // Extrapolation using curvature
        cl = Lift(sz,1) + gradn*(alph-Lift(sz,0)) + pow(alph-Lift(sz,0),2)/2*curvn;
    }else {
        // linear interpolation
        cl = Lift(i-1,1) + (Lift(i,1)-Lift(i-1,1))/(Lift(i,0)-Lift(i-1,0))*(alph-Lift(i-1,0));
    }

    return cl;
}

double inputData::getCD(double a /*Radians*/){
    double alph = a/pi*180;
```

```
    double cd;
    int i = 0;
    int sz = Drag.rows()-1;

    if (alph > 50) alph = 50;
    if (alph < -50) alph = -50;

    while (i < Drag.rows()) {
        if (alph <= Drag(i,0))
        {
            break;
        }
        i++;
    }

    // If a is outside of data bounds
    // 2nd order extrapolation

    double grad0 = (Drag(1,1)-Drag(0,1))/(Drag(1,0)-Drag(0,0));
    double grad0_1 = (Drag(2,1)-Drag(0,1))/(Drag(2,0)-Drag(0,0));

    double gradn = (Drag(sz,1)-Drag(sz-1,1))/(Drag(sz,0)-Drag(sz-1,0));
    double gradn_1 = (Drag(sz,1)-Drag(sz-2,1))/(Drag(sz,0)-Drag(sz-2,0));

    double curv0 = (grad0_1-grad0)/(Drag(1,0)-Drag(0,0));
    double curvn = (gradn-gradn_1)/(Drag(sz,0)-Drag(sz-1,0));

    if (i==0) {
        // Extrapolation using curvature
        cd = Drag(0,1) + grad0*(alph-Drag(0,0)) + pow(alph-Drag(0,0),2)/2*curv0;
    } else if (i == Drag.rows()) {
        // Extrapolation using curvature
        cd = Drag(sz,1) + gradn*(alph-Drag(sz,0)) + pow(alph-Drag(sz,0),2)/2*curvn;
    }else {
        cd = Drag(i-1,1) + (Drag(i,1)-Drag(i-1,1))/(Drag(i,0)-Drag(i-1,0))*(alph-Drag(i-1,0));
    }
    return cd;
}



double inputData::getCDgrad(double a /*Radians*/){

    double grad;
    double step = 0.1/180.0*pi;

    grad = (getCD(a+step)-getCD(a-step))/(2*step);

    return grad;
}

double inputData::getCLgrad(double a /*Radians*/){
    double grad;

    double step = 0.1/180.0*pi;

    grad = (getCL(a+step)-getCL(a-step))/(2*step);

    return grad;
}

double inputData::getAeff(){
    double aeff;
    aeff = -atan((U_inf*sin(alpha+theta)-wv)/(-U_inf*cos(alpha+theta)-uv))-theta;
    return aeff;
}

void inputData::initialize(){
    calcDamp();
    computeEigen();
    calcSteadyLoads();
    // Calculate Partial solution
    partialSol = K.householderQr().solve(Q);
```

```
}
```

I.iii Αρχείο συναρτήσεων επίλυσης solver.cpp

```
#include "utilities.h"


void inputData::computeEigen(){

    // Initialize help matrices
    Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> R;
    Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> newR;

    Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> L;
    Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> invL;

    R.resize(2*siz, 2*siz);
    newR.resize(2*siz, 2*siz);
    L.resize(2*siz, 2*siz);
    invL.resize(2*siz, 2*siz);

    if(theodor)
    {
        C.setZero();
    }

    // Construct A Matrix
    R.setZero(); L.setIdentity();
    R.block(0,0,2,2) = -C;
    R.block(0,2,2,2) = -K;
    R(2,0) = 1;
    R(3,1) = 1;

    L(0,0) = M(0,0); L(1,1) = M(1,1);
    invL = L.colPivHouseholderQr().inverse();

    newR = invL*R;

    // Compute eigenvalues of A
    Eigen::EigenSolver<Eigen::MatrixXd> ces;
    ces.setMaxIterations(500);
    ces.compute(newR, true);

    // Save eigenvectors in data class
    for (int i =0;i<4;i++)
    {
        eigenvectors(i,0).set(ces.eigenvectors()(i,0));
        eigenvectors(i,1).set(ces.eigenvectors()(i,2));
    }

    eigenvalues[0] = ces.eigenvalues()[0];
    eigenvalues[1] = ces.eigenvalues()[2];

}

// Solves for the homogenous solution coefficients
// with boundary conditions as input
void inputData::computeConstants() {

    Eigen::Matrix<double,4,4> coeff;
    Eigen::Matrix<double,4,1> rhs;
    Eigen::Matrix<double,4,1> temp;
```

```
    coeff.setZero();

    // Solves x=x_0 for t0 system of equations
    for (int i = 0;i<4;i++)
    {
        coeff(i,0) = eigenvectors(i,0).len*cos(eigenvectors(i,0).theta);
        coeff(i,1) = eigenvectors(i,0).len*sin(eigenvectors(i,0).theta);
        coeff(i,2) = eigenvectors(i,1).len*cos(eigenvectors(i,1).theta);
        coeff(i,3) = eigenvectors(i,1).len*sin(eigenvectors(i,1).theta);
    }

    // Set boundary conditions as rhs
    rhs(0,0) = uv;
    rhs(1,0) = wv;
    rhs(2,0) = u;
    rhs(3,0) = w;

    // Calculate Partial solution
    if (theodor)
    {
        partialSol = K.householderQr().solve(Q);
    }

    // Add partial solution to the rhs
    rhs(2,0) -= partialSol(0);
    rhs(3,0) -= partialSol(1);

    // Solve system -> obtain coefficients
    temp = coeff.householderQr().solve(rhs);
    A1 = temp(0,0);
    B1 = temp(1,0);
    A2 = temp(2,0);
    B2 = temp(3,0);

    // Output state data for plotting
    std::ofstream out("state.dat");

    double ang, x1, x2, y1, y2;

    for (double t = 0; t<360; t ++)
    {
        ang = t*pi/180;
        x1 = eigenvectors(2,0).len*(A1*cos(ang+eigenvectors(2,0).theta) + B1*sin(ang+eigenvectors(2,0).theta));
        x2 = eigenvectors(2,1).len*(A2*cos(ang+eigenvectors(2,1).theta) + B2*sin(ang+eigenvectors(2,1).theta));
        y1 = eigenvectors(3,0).len*(A1*cos(ang+eigenvectors(3,0).theta) + B1*sin(ang+eigenvectors(3,0).theta));
        y2 = eigenvectors(3,1).len*(A2*cos(ang+eigenvectors(3,1).theta) + B2*sin(ang+eigenvectors(3,1).theta));
        out << x1 << " " << y1 << " " <<  x2 << " " << y2 << std::endl;
    }
    out.close();
}
// Newmark integration (Unused)
void inputData::calcNextStep(){

    // Newmark integration method

    // Newmark constants
    double beta = 1.0/4.0;
    double Gamma = 0.5;

    Eigen::Matrix<double, 2, 2> Keff;
    Eigen::Vector<double, 2> Qeff, x, xv, temp;

    t_tot += dt;

    // Intermediate predictor values
    x(0) = u + dt*uv + (0.5-beta)*pow(dt,2)*ua;
    x(1) = w + dt*wv + (0.5-beta)*pow(dt,2)*wa;

```

```
    xv(0) = uv + (1-Gamma)*dt*ua;
    xv(1) = wv + (1-Gamma)*dt*wa;

    // Construct Effective stiffness and load matrices
    Keff = M/(beta*pow(dt,2)) + C*Gamma/(beta*dt) + K;

    Qeff = Q + (M/(beta*pow(dt,2)) + Gamma*C/(beta*dt))*x - C*xv;

    temp = Keff.householderQr().solve(Qeff);

    u = temp(0);
    w = temp(1);

    // Update velocity and accel values
    uv = xv(0) + Gamma/(beta*dt)*(u-x(0));
    wv = xv(1) + Gamma/(beta*dt)*(w-x(1));

    ua = (u-x(0))/(beta*dt);
    wa = (w-x(1))/(beta*dt);


    std::ofstream out("time.dat", std::ofstream::app);

     out << t_tot << std::setw(25) << u << std::setw(25) << w << std::setw(25) << uv << std::setw(25) << wv << std::setw(25) << ua << std::setw(25) << wa << std::endl;
    out.close();

}

void inputData::nextAnalyticalStep(std::string filename) {

    std::vector<double> var(4);

    t_tot += dt;

    // Calculate homogenous solution for each variable
    for (int i = 0; i<4; i++)
    {
        var[i] = eigenvectors(i,0).len*std::exp(eigenvalues[0].real()*dt)*(A1*cos(eigenvalues[0].imag()*dt+eigenvectors(i,0).theta)
                 + B1*sin(eigenvalues[0].imag()*dt + eigenvectors(i,0).theta))

                + eigenvectors(i,1).len*std::exp(eigenvalues[1].real()*dt)*(A2*cos(eigenvalues[1].imag()*dt+eigenvectors(i,1).theta)
                + B2*sin(eigenvalues[1].imag()*dt + eigenvectors(i,1).theta));

    }


    // Add partial solution
    var[2] += partialSol(0);
    var[3] += partialSol(1);

    // Calculate acceleration values (FD)
    ua = (var[0]-uv)/dt;
    wa = (var[1]-wv)/dt;

    // Update solution state variables
    u = var[2]; w = var[3]; uv = var[0]; wv = var[1];

    // File output
    std::ofstream outAn(filename, std::ofstream::app);

    outAn << t_tot << std::setw(25) << u << std::setw(25) << w << " " << uv << " " << wv << " " << ua << " " << wa << std::endl;

    outAn.close();
}
```

I.iv Αρχείο συναρτήσεων εξισώσεων μόνιμης ροής steady.cpp

```cpp
#include "utilities.h"

// Calculates steady damping matrix
void inputData::calcDamp() {
    double partaU, partaW, aeff;

    // Aeff partial derivatives

    partaU = - (U_inf*sin(alpha+theta)-wv)/(pow(U_inf,2) + pow(uv,2) + pow(wv,2) + 2*U_inf*(cos(alpha+theta)*uv - sin(alpha+theta)*wv));
    partaW = - (U_inf*cos(alpha+theta)+uv)/(pow(U_inf,2) + pow(uv,2) + pow(wv,2) + 2*U_inf*(cos(alpha+theta)*uv - sin(alpha+theta)*wv));
    aeff = -atan((U_inf*sin(alpha+theta)-wv)/(-U_inf*cos(alpha+theta)-uv))-theta;

    // Construction of C Matrix
    C(0,0) = -0.5*rho*chord*pow(U_inf,2)*partaU * (sin(aeff+theta)*(getCLgrad(aeff)+getCD(aeff)) + cos(aeff+theta)*(getCL(aeff)-getCDgrad(aeff)) );

    C(0,1) = -0.5*rho*chord*pow(U_inf,2)*partaW * (sin(aeff+theta)*(getCLgrad(aeff)+getCD(aeff)) + cos(aeff+theta)*(getCL(aeff)-getCDgrad(aeff)) );

    C(1,0) = -0.5*rho*chord*pow(U_inf,2)*partaU * (sin(aeff+theta)*(getCDgrad(aeff)-getCL(aeff)) + cos(aeff+theta)*(getCD(aeff)+getCLgrad(aeff)) );

    C(1,1) = -0.5*rho*chord*pow(U_inf,2)*partaW * (sin(aeff+theta)*(getCDgrad(aeff)-getCL(aeff)) + cos(aeff+theta)*(getCD(aeff)+getCLgrad(aeff)) );

    // File output
    std::ofstream dampSt("dampingSteady.dat", std::ofstream::app);
    dampSt << t_tot << std::setw(25) << C(0,0) << std::setw(25) << C(0,1) << std::setw(25) << C(1,0) << std::setw(25) << C(1,1) << std::endl;
    dampSt.close();
}

// Calculates steady aerodynamic loads
void inputData::calcSteadyLoads() {

    double aeff = getAeff();
    double L, D;
    L = 0.5*getCL(aeff)*rho*pow(U_inf,2)*chord;
    D = 0.5*getCD(aeff)*rho*pow(U_inf,2)*chord;

    Q(0) = L*sin(aeff+theta)-D*cos(aeff+theta);
    Q(1) = L*cos(aeff+theta)+D*sin(aeff+theta);

    // File output

    std::ofstream loadsSt("loadsSteady.dat", std::ofstream::app);
    loadsSt << t_tot << std::setw(25) << Q(0) << std::setw(25) << Q(1) << std::endl;
    loadsSt.close();

}
```

I.v Αρχείο συναρτήσεων εξισώσεων μη-μόνιμης ροής theodorsen.cpp

```cpp
#include "utilities.h"

void Theodorsen::calcNextY() {
    for (int i = 0; i<2; i++) {
```

```
        y[i] = (b[i]*A[i]*2*target->U_inf/target->chord*target->getAeff()+y[i]/dt)/(1/dt+b[i]*2*target->U_inf/target->chord);
    }
}

// Calculate unsteady aerodynamic loads
void Theodorsen::calcLoads() {

    ae = target->getAeff()*(1-A[0]-A[1])+y[0]+y[1];

    double h_dd = target->wa*cos(target->theta) + target->ua*sin(target->theta); // Flapwise acceleration

    double cl = grad0*(ae-a0) - pi*target->chord*h_dd/(2*pow(target->U_inf,2));
    double cd = target->getCD(ae) + cl*(target->getAeff()-ae);

    L = 0.5*target->rho*pow(target->U_inf,2)*cl*target->chord;
    D = 0.5*target->rho*pow(target->U_inf,2)*cd*target->chord;

    target->Q(0) = L*sin(ae+target->theta)-D*cos(ae+target->theta);
    target->Q(1) = L*cos(ae+target->theta)+D*sin(ae+target->theta);

    // File output
    std::ofstream loadsTh("loadsTheodorsen.dat", std::ofstream::app);
    loadsTh << target->t_tot << std::setw(25) << target->Q(0) << std::setw(25) << target->Q(1) << std::endl;
    loadsTh.close();

}

void Theodorsen::calcDamp() {

    // Theodorsen derivation Aeff

    // Aeff partial derivatives
    double partaU = - (target->U_inf*sin(target->alpha+target->theta)-target->wv)/(pow(target->U_inf,2) + pow(target->uv,2) + pow(target->wv,2) + 2*target->U_inf*(cos(target->alpha+target->theta)*target->uv - sin(target->alpha+target->theta)*target->wv));

    double partaW = - (target->U_inf*cos(target->alpha+target->theta)+target->uv)/(pow(target->U_inf,2) + pow(target->uv,2) + pow(target->wv,2) + 2*target->U_inf*(cos(target->alpha+target->theta)*target->uv - sin(target->alpha+target->theta)*target->wv));


    // Cl, Cd partial derivatives
    double partClU = grad0*partaU;
    double partClW = grad0*partaW;

    double partCdU = target->getCDgrad(ae)*partaU + partClU*(target->getAeff()-ae);
    double partCdW = target->getCDgrad(ae)*partaW + partClW*(target->getAeff()-ae);

    // Lift, Drag derivatives
    double partLU = 0.5*target->rho*pow(target->U_inf,2)*target->chord*partClU;
    double partLW = 0.5*target->rho*pow(target->U_inf,2)*target->chord*partClW;

    double partDU = 0.5*target->rho*pow(target->U_inf,2)*target->chord*partCdU;
    double partDW = 0.5*target->rho*pow(target->U_inf,2)*target->chord*partCdW;

    // C matrix assembly
    target->C(0,0) = partLU*sin(ae+target->theta) + L*cos(ae+target->theta)*partaU
                   - partDU*cos(ae+target->theta) + D*sin(ae+target->theta)*partaU;

    target->C(0,1) = partLW*sin(ae+target->theta) + L*cos(ae+target->theta)*partaW
                   - partDW*cos(ae+target->theta) + D*sin(ae+target->theta)*partaW;

    target->C(1,0) = partLU*cos(ae+target->theta) - L*sin(ae+target->theta)*partaU
                   + partDU*sin(ae+target->theta) + D*cos(ae+target->theta)*partaU;

    target->C(1,1) = partLW*cos(ae+target->theta) - L*sin(ae+target->theta)*partaW
                   + partDW*sin(ae+target->theta) + D*cos(ae+target->theta)*partaW;

    target->C = -target->C;

```

```
    // File output
    std::ofstream dampTh("dampingTheodorsen.dat", std::ofstream::app);
    dampTh << target->t_tot << std::setw(25) << target->C(0,0) << std::setw(25) << target->C
    (0,1) << std::setw(25) << target->C(1,0) << std::setw(25) << target->C(1,1) << std::endl;
    dampTh.close();

}
```

I.vi Βοηθητικό αρχείο ορισμού συναρτήσεων utilities.h

```
#pragma once

//#include <Eigen/src/Core/util/Constants.h>
#include <complex>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <Eigen/Core>
#include <Eigen/Eigenvalues>
#include <string>
#include <numbers>
#include <vector>
#include <iomanip>

#define dt 0.005

const int parInitVal = 25;
const double pi = std::numbers::pi;

struct cmpx {
    double len, theta, real, imag;
    void set(std::complex<double> val) {
        len = sqrt(pow(val.real(),2) + pow(val.imag(),2));
        theta = atan2(val.imag(), val.real());
        real = val.real();
        imag = val.imag();
    }
};

struct inputData {
    // Struct variables
    int siz;
    double alpha, theta, U_inf, rho, chord, A1, A2, B1, B2, t_tot;
    Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> M, K, C;
    Eigen::Matrix<double, Eigen::Dynamic, 2> Lift, Drag;
    Eigen::Matrix<cmpx,4,2> eigenvectors;
    Eigen::Matrix<cmpx,4,2> prevEigenvectors;
    Eigen::Vector<double,2> Q;
    Eigen::Vector<double,2> prevQ;
    Eigen::Vector<double,2> partialSol;

    std::vector<std::complex<double> > eigenvalues;
    std::vector<std::complex<double> > prevEigenvalues;

    // Methods declaration
    double getCD(double a);
    double getCDgrad(double a);
    double getCL(double a);
    double getCLgrad(double a);
    void readInput(std::string filename);
    void initialize();
    void calcDamp();
```

```
    void computeEigen();
    void computeConstants();
    void calcNextStep();
    void calcSteadyLoads();
    void nextAnalyticalStep(std::string filename);
    double getAeff();


    // State variables
    double u, uv, ua, w, wv, wa;

    bool theodor; // Unsteady solution flag (True)

    inputData(int sz)
    {
        siz = sz;
        M.resize(sz,sz);
        K.resize(sz,sz);
        C.resize(sz,sz);
        M.setZero(); K.setZero(); C.setZero();
        Lift.resize(parInitVal,2);
        Drag.resize(parInitVal,2);
        eigenvalues.resize(2);
        prevEigenvalues.resize(2);
        prevQ.setZero();
        u = 0;
        uv = 0;
        ua = 0;
        w = 0;
        wv = 0;
        wa = 0;
        t_tot = 0;
        Q.setZero();
        theodor = false;
    }
};

// Struct used to compute unsteady variants
struct Theodorsen {
    inputData *target;
    double b[2];
    double A[2];
    double y[2];
    double ae, L, D;
    double grad0, a0;

    void calcNextY();
    void calcDamp();
    void calcLoads();
    // Constructor assigns target struct pointer
    Theodorsen(inputData* p) : target(p){
        b[0] = 0.0455;
        b[1] = 0.3;
        A[0] = 0.165;
        A[1] = 0.335;
        if (target->t_tot != 0) {
            std::cout << "Instantiated Theodorsen on non-initial solution" << std::endl;
        }else {
            y[0] = target->getAeff()*A[0];
            y[1] = target->getAeff()*A[1];
        }

        grad0 = (target->getCL(0.0+5.0*pi/180.0)-target->getCL(0.0-5.0*pi/180.0))/((double)
    2.0*5.0/180.0*pi);

        if (grad0 != 0)
        {
            a0 = -target->getCL(0)/grad0;
        }else {
            std::cout << "zero CL gradient" << std::endl;
            a0 = 0;
        }
    };
```

```
};

void readAero(std::string filenameCl, std::string filenameCd, inputData &input);

void readMat(int sz, Eigen::Matrix<double,Eigen::Dynamic,Eigen::Dynamic> &M, Eigen::Matrix<
    double,Eigen::Dynamic,Eigen::Dynamic> &K, std::string filename);
```

I.vii Αρχεία εισόδου matrix.dat και input.dat

```
UINF (m/s)
80
THETA_0 (Degrees)
2
AoA (Degrees)
4
Density
1.22
Chord Length (m)
1.5
u0
0
uv0
0
ua0
0
w0
0
wv0
0
wa0
0
```

```
M
165 0
0   165
K
15791   0
0       3948
```