ΕΛΛΗΝΙΚΟ
ΑΝΟΙΚΤΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΕΣ ΣΠΟΥΔΕΣ ΣΤΑ ΜΑΘΗΜΑΤΙΚΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήση Physics Informed Neural Networks για την επίλυση γραμμικών και μη γραμμικών προβλημάτων φυσικής.

ΚΟΛΛΙΑΣ ΝΙΚΟΛΑΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ Α': ΜΑΤΖΑΚΟΣ ΝΙΚΟΛΑΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ Β': ΤΣΙΤΣΑΣ ΝΙΚΟΛΑΟΣ

ΑΘΗΝΑ

Μάιος,2026

Περίληψη

Στην παρούσα εργασία χρησιμοποιούμε τα νευρωνικά δίκτυα ενημερωμένα με εξισώσεις φυσικής (Physics Informed Neural Networks ή PINNs) για να επιλύσουμε μη γραμμικά προβλήματα της φυσικής. Σαν πρότυπο πρόβλημα επιλέχθηκε το χαοτικό πρόβλημα των τριών σωμάτων της ουράνιας μηχανικής. Η κωδικοποίηση έγινε σε γλώσσα Python με χρήση της βιβλιοθήκης DeepXDE. Τα scripts που χρησιμοποιήθηκαν στην εργασία έχουν αναρτηθεί στο github.

Το 1ο κεφάλαιο περιέχει μια σύντομη περιγραφή των βασικών εννοιών των νευρωνικών δικτύων με έμφαση στα νευρωνικά δίκτυα που χρησιμοποιούνται στην εργασία. Στο 2ο κεφάλαιο περιγράφονται τα νευρωνικά δίκτυα ενημερωμένα με εξισώσεις φυσικής (PINNs). Το 3ο κεφάλαιο περιέχει μια σύντομη περιγραφή της βιβλιοθήκης DeepXDE. Στο 4ο κεφάλαιο περιγράφονται τα πειράματα με την βιβλιοθήκη DeepXDE καθώς και οι τεχνικές που χρησιμοποιήθηκαν για την προσέγγιση των τροχιών. Αρχικά εκπαιδεύουμε τα Pinns χωρίς εξωτερικά δεδομένα. Ξεκινάμε με το πρόβλημα των δύο σωμάτων που είναι αναλυτικά επιλύσιμο και κατόπιν ασχολούμαστε με το μη επιλύσιμο και χαοτικό πρόβλημα των τριών σωμάτων. Συγκεκριμένα προσεγγίζουμε τρεις διάσημες *περιοδικές λύσεις* του προβλήματος των τριών σωμάτων, την λύση του *Euler*, την λύση του *Lagrange* και την πιο σύγχρονη περιοδική τροχιά σχήματος οκτώ (*figure eight*). Ιδιαίτερη έμφαση δίνεται στην γενίκευση της λύσης από το νευρωνικό δίκτυο και στον εντοπισμό περιπτώσεων ελλιπούς (ή και αποτυχημένης) εκπαίδευσης του δικτύου. Για παράδειγμα μια πολύ καλή οπτικά τροχιά μπορεί να είναι απλά μια υπερεκπαίδευση στα σημεία ελέγχου και να αποτυγχάνει σε ενδιάμεσα σημεία που δεν εκπαιδεύτηκε το δίκτυο. Κατόπιν συγκρίνουμε τα νευρωνικά δίκτυα με τα απλά νευρωνικά δίκτυα στην περίπτωση *πειραματικών δεδομένων με υψηλό θόρυβο*. Η σύγκριση γίνεται σε δύο περιπτώσεις: α) τα Pinns γνωρίζουν τις αρχικές συνθήκες του προβλήματος και β) τα pinns δεν γνωρίζουν τις αρχικές συνθήκες. Στην δεύτερη περίπτωση παρατηρήσαμε ότι η έλλειψη αρχικών συνθηκών οδήγησε τα pinns να βρουν αρχικές συνθήκες που αντιστοιχούν σε περιοδικές λύσεις που δεν δίνονταν από τα πειραματικά δεδομένα. Αυτήν την ενδιαφέρουσα ιδιότητα εξερευνήσαμε στην τελευταία παράγραφο. Συγκεκριμένα δίνοντας δεδομένα από γνωστές περιοδικές τροχιές σχήματος οκτώ τα pinns έδωσαν αρχικές συνθήκες κοντά σε διαφορετικές περιοδικές τροχιές τύπου ΒΗΗ (από τα αρχικά των ερευνητών που τις ανακάλυψαν). Για τις ακριβείς περιοδικές συνθήκες χρησιμοποιήθηκε μια μέθοδος ελαχίστων τετραγώνων που αναζητά περιοδικές τροχιές κοντά στις αρχικές συνθήκες που έδωσε το pinn. Τέλος επεκτείναμε την μέθοδο σε *αφύσικα* δεδομένα με υψηλό θόρυβο που όμως είναι περιορισμένα σε κάποια "μικρή" περιοχή

του χώρου με τροχιές κυκλικές ή ελλειπτικές και πήραμε αρκετές γνωστές περιοδικές λύσεις του προβλήματος των τριών σωμάτων.

Στο 5ο κεφάλαιο αναφέρουμε τα συμπεράσματα από τα πειράματα και γίνεται μια σύντομη συζήτηση για τα πλεονεκτήματα/μειονεκτήματα των Pinns σε σχέση με τις παραδοσιακές αριθμητικές μεθόδους.

Πίνακας περιεχομένων

1 Τεχνητά Νευρωνικά Δίκτυα (ANNs)

1.1 Ιστορική αναδρομή

Από την αρχαιότητα οι άνθρωποι ενδιαφέρθηκαν για τους μηχανισμούς της μνήμης και της μάθησης. Από τους πρώτους που έκαναν συστηματική προσπάθεια είναι ο Αριστοτέλης (300 π.χ) που διατύπωσε τους νόμους των συνειρμών. Την θεωρία των συνειρμών (Associationism) μελέτησαν και επέκτειναν αργότερα και άλλοι φιλόσοφοι και ψυχολόγοι όπως οι Thomas Hobbes (1588-1679) , John Locke (1632-1704), David Hume (1711-1776) και άλλοι. (Wang & Raj, 2017)

Παρ όλα αυτά η κατανόηση της δομής και της λειτουργίας του εγκεφάλου έγιναν γνωστές πολύ αργότερα. Ο Alexander Bain το 1873, με αφορμή τις ανακαλύψεις της νευροανατομίας ήταν ο πρώτος που πρότεινε την υπόθεση ότι ο ανθρώπινος εγκέφαλος αποτελείται από διασυνδεδεμένους νευρώνες και ότι κάθε εγκεφαλική δράση σχετίζεται με ένα συγκεκριμένο σύνολο νευρώνων και συνδέσεων με γειτονικούς νευρώνες. Έκανε επίσης την πρώτη διατύπωση του «κανόνα των συσχετίσεων», στον οποίο δηλώνει ότι η επαναλαμβανόμενη εμφάνιση της ίδιας σκέψης, αίσθησης ή εντύπωσης τείνει να διεγείρει τις συνδέσεις μεταξύ νευρώνων που σχετίζονται με αυτές τις ενέργειες, αυξάνοντας τον βαθμό σύνδεσής τους ή μειώνοντας την παρεμπόδιση αυτών των ενεργειών.

Το 1943, οι Warren McCulloch και Walter Pitts παρουσίασαν το πρώτο τεχνητό νευρωνικό δίκτυο, του οποίου η λειτουργία βασίστηκε στο μοντέλο του βιολογικού νευρώνα.

Το 1949 ο Donald Hebb επισημοποίησε τον κανόνα μάθησης για αλλαγές στις συναπτικές συνδέσεις . Ο συγγραφέας πρότεινε ότι εάν δύο νευρώνες είναι συνδεδεμένοι μεταξύ τους και ο πρώτος επιμένει επανειλημμένα να ενεργοποιεί τον δεύτερο, τότε η συναπτική τιμή της σύνδεσης μεταξύ των δύο θα πρέπει να αυξηθεί.

Το 1958, ο ψυχολόγος Frank Rosenblatt ανέπτυξε το Perceptron ένα μοντέλο τεχνητού νευρώνα που χρησιμοποιείται και σήμερα με μικρές αλλαγές. Επίσης ανέπτυξε έναν κανόνα μάθησης ώστε το σύστημά του να μπορεί να κάνει ταξινόμηση μοτίβων. Το δίκτυο που χρησιμοποίησε είχε ένα μόνο κρυφό στρώμα (hidden layer) και θεωρείται ο πρόγονος των σημερινών πολυεπίπεδων νευρωνικών δικτύων. (Pires κ.ά., 2023)

Μετά από την αρχική περίοδο ενθουσιασμού ακολούθησε μια περίοδος με μειωμένο ενδιαφέρον. Το 1969 οι Marvin Minsky και Seymour Papert έγραψαν το βιβλίο *Perceptrons: An Introduction to*

Computational Geometry όπου παρουσίασαν τους περιορισμούς που έχουν τα perseptrons ενός στρώματος. Αν και οι συγγραφείς γνώριζαν ότι τα ισχυρά perceptron έχουν πολλαπλά στρώματα και τα βασικά perceptron του Rosenblatt έχουν τρία στρώματα, όρισαν το perceptron ως μια μηχανή δύο επιπέδων που μπορεί να χειριστεί μόνο γραμμικά διαχωρίσιμα προβλήματα . Για παράδειγμα, δεν μπορεί να λύσει το πρόβλημα αποκλειστικής OR (XOR).

Την περίοδο αυτή λόγω του μικρού ενδιαφέροντος και της ελάχιστης χρηματοδότησης μόνο λίγοι ερευνητές συνέχισαν να εργάζονται στο αντικείμενο. Παρ όλα αυτά έγινε πρόοδος σε θέματα που ακόμα και σήμερα αποτελούν αντικείμενο έρευνας. Ο Paul Werbos το 1974 πρότεινε τη μέθοδο αναδρομικής διάδοσης (back-propagation) για την εκπαίδευση των νευρωνικών δικτύων όμως η σημασία της εκτιμήθηκε πλήρως μετά το 1986.

Την δεκαετία του 1980 το ενδιαφέρον για τα νευρωνικά δίκτυα ανανεώθηκε. Ο Teuvo Kohonen εισήγαγε το τεχνητό νευρωνικό δίκτυο που μερικές φορές ονομάζεται χάρτης ή δίκτυο Kohonen. Το 1982 ο John Joseph Hopfield περιγράφει το Hopfield-network έναν τύπο αναδρομικού νευρωνικού δικτύου (recurrent network) που χρησιμοποιείται κυρίως ως σύστημα μνήμης με προσπέλαση βάσει περιεχομένου. Ο αλγόριθμος backpropagation, που προτάθηκε αρχικά από τον Werbos το 1974, έγινε ευρέως γνωστός το 1986 με το βιβλίο *Learning Internal Representation by Error Propagation* των *Rumelhart, Hinton και Williams*. Ο backpropagation είναι μια μορφή αλγορίθμου gradient descent που χρησιμοποιείται στα τεχνητά νευρωνικά δίκτυα για ελαχιστοποίηση της συνάρτησης σφάλματος και την ενημέρωση των βαρών(weights) του δικτύου. (Macukow, 2016)

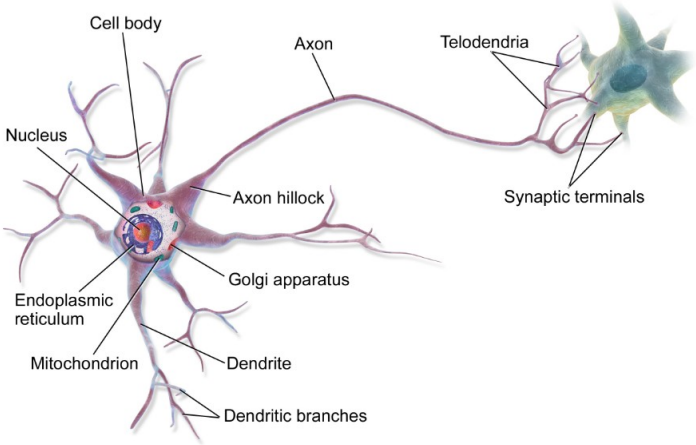
Για μια λεπτομερή ιστορική αναδρομή παραπέμπουμε στην τεχνική αναφορά "*Annotated History of Modern AI and Deep Learning*" του *Jürgen Schmidhuber.* (Schmidhuber, 2022)

Σήμερα το ενδιαφέρον για τα νευρωνικά δίκτυα είναι μεγαλύτερο από ποτέ. Η μεγάλη υπολογιστική ισχύ και μνήμη των σύγχρονων υπολογιστικών συστημάτων σε συνδυασμό με την ανάπτυξη μοντέλων τεχνητής νοημοσύνης που μπορούν να κατανοούν την ανθρώπινη γλώσσα (Large Language Systems-LLMs) και να δημιουργούν νέο περιεχόμενο (generative AI) έκανε δυνατή την ανάπτυξη συστημάτων όπως το ChatGPT ,Gemini κ.α με τεράστιο εμπορικό ενδιαφέρον. Αλλά και στον επιστημονικό τομέα το ενδιαφέρον είναι πολύ μεγάλο αφού σήμερα χρησιμοποιούνται βαθιά νευρωνικά δίκτυα στην Ιατρική, στην Βιολογία, στην πρόβλεψη καιρικών φαινομένων, και σχεδόν σε όλες τις περιοχές επιστημονικού ενδιαφέροντος.

1.2 Ο Βιολογικός νευρώνας

Η βασική λειτουργική μονάδα του κεντρικού νευρικού συστήματος και επομένως και του εγκεφάλου είναι ο νευρώνας. Οι νευρώνες είναι κύτταρα που μπορούν να συνδέονται με άλλους νευρώνες δημιουργώντας ένα εξαιρετικά πολύπλοκο δίκτυο. Ο ανθρώπινος εγκέφαλος έχει κατά μέσο όρο 100 δισεκατομμύρια νευρώνες. Κάθε νευρώνας μπορεί να συνδεθεί με μέχρι 10000 άλλους νευρώνες. Έτσι στον ανθρώπινο εγκέφαλο έχουμε τον αστρονομικό αριθμό των 1000 τρισεκατομμυρίων συνδέσεων (συνάψεων). Η αποθηκευτική ικανότητα του εγκεφάλου (μνήμη) είναι πολύ δύσκολο να υπολογισθεί. Υπάρχουν εκτιμήσεις που κυμαίνονται από 1 μέχρι 1000 Terabyte.

Οι νευρώνες εξειδικεύονται ώστε να επεξεργάζονται χημικά σήματα που δέχονται από τους άλλους νευρώνες με τους οποίους έχουν συνάψεις. Μετά από την επεξεργασία αυτών των σημάτων ο νευρώνας "αποφασίζει" αν θα προωθήσει το σήμα σε άλλους νευρώνες με τους οποίους συνδέεται. Η προώθηση αυτή γίνεται μέσω μίας ηλεκτροχημικής διεργασίας.

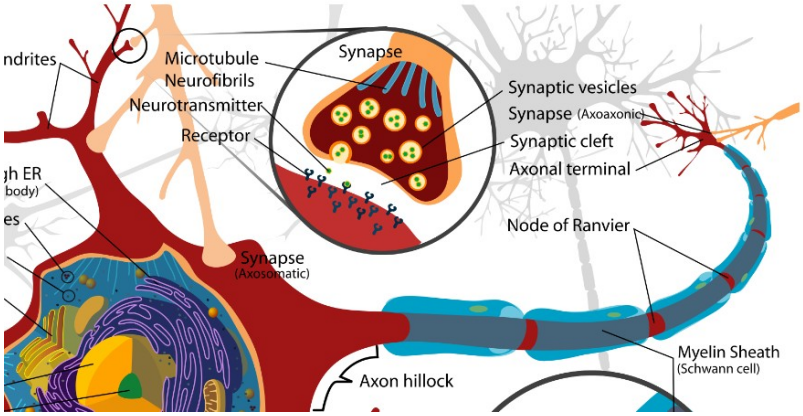


Σχήμα 1: Αναπαράσταση νευρώνα εγκεφάλου (Zhang, 2019)

Στο σχήμα 1 έχουμε την αναπαράσταση ενός πολυπολικού (Multipolar) νευρώνα που συναντάται στον εγκέφαλο. Παρατηρώντας από αριστερά προς τα δεξιά διακρίνουμε τα παρακάτω βασικά μέρη:

- **Δενδρίτες** (Dendrites): είναι αποφυάδες με πολλές διακλαδώσεις (Dendritic Branches). Οι δενδρίτες δέχονται τα εισερχόμενα (χημικά) σήματα από άλλους νευρώνες.

- **Σώμα** (cell body): είναι το κυρίως μέρος του νευρώνα όπου βρίσκεται ο κυτταρικός πυρήνας και τα οργανίδια του κυττάρου (μιτοχόνδρια κ.α). Εδώ γίνεται η επεξεργασία των εισερχόμενων σημάτων.
- **Νευρικός άξονας** ή νευρίτης (Axon): όταν ο νευρώνας ενεργοποιηθεί ένας ηλεκτρικός παλμός διαδίδεται κατά μήκος του νευρικού άξονα. Στο τέλος του ο άξονας έχει πολλές διακλαδώσεις (Telodendria) ώστε να μπορεί να συνδεθεί με τους επόμενους νευρώνες. Οι περισσότεροι νευρώνες έχουν έναν μόνο νευρικό άξονα.
- **Τελικές απολήξεις** ή τελικά κομβία (Axon Terminals): στο άκρο του νευρικού άξονα υπάρχουν οι τελικές απολήξεις που περιέχουν τις συνάψεις. Στις τελικές απολήξεις υπάρχουν δομές που ονομάζονται τελικά κομβία και απελευθερώνουν τους νευροδιαβιβαστές στα κύτταρα στόχους. Οι νευροδιαβιβαστές είναι χημικές ουσίες που μπορούν να δράσουν είτε διεγερτικά είτε ανασταλτικά στα κύτταρα-στόχους.



Σχήμα 2: Σύναψη (Zhang, 2019)

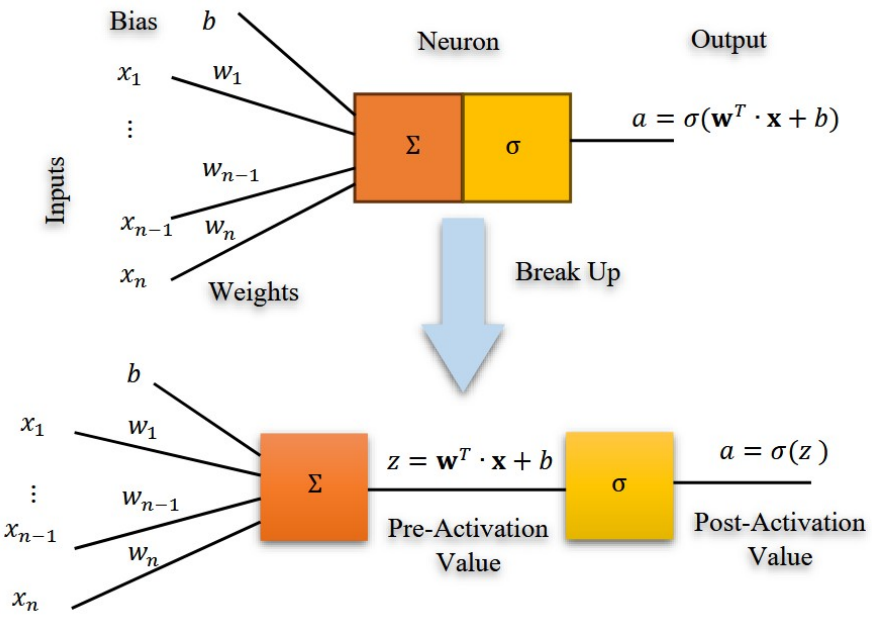
Στο σχήμα 2 φαίνεται μια αναπαράσταση της σύναψης (Synapse) όπου απελευθερώνονται νευροδιαβιβαστές (Neurotransmitters). Ο νευρώνας-στόχος έχει κατάλληλους υποδοχείς (Receptors) που δέχονται τους νευροδιαβιβαστες και μεταφέρουν με χημικό τρόπο το σήμα. (Zhang, 2019)

Τέλος είναι σημαντικό να αναφέρουμε ότι οι νευρώνες του εγκεφάλου έχουν την ικανότητα να τροποποιούν τις συνάψεις τους ή να δημιουργούν νέες συνάψεις ανάλογα με τα ερεθίσματα που δέχονται από το περιβάλλον τους. Αυτή η ικανότητα γνωστή ως πλαστικότητα (Neuroplasticity) παίζει καίριο ρόλο στους μηχανισμούς της μάθησης και της μνήμης.(Marzola κ.ά., 2023)

1.3 Ο τεχνητός νευρώνας

Ο τεχνητός νευρώνας είναι ένα υπεραπλουστευμένο μοντέλο του βιολογικού νευρώνα. Είναι η βασική μονάδα ενός τεχνητού νευρωνικού δικτύου (ANN -Artificial Neural Networks) και αποτελείται από n εισόδους και μία μοναδική έξοδο. Κάθε είσοδος έχει ένα βάρος w που μοντελοποιεί την ισχύ της σύναψης. Επίσης συνήθως έχουμε και μία είσοδο με σταθερή τιμή ίση με 1 και βάρος b που αποκαλείται bias.

Για να υπολογίσουμε την έξοδο ενός τεχνητού νευρώνα, αρχικά υπολογίζουμε το σταθμισμένο άθροισμα των εισόδων, το οποίο ονομάζεται τιμή προ-ενεργοποίησης (pre-activation value). Στη συνέχεια εφαρμόζουμε τη συνάρτηση ενεργοποίησης (activation function) για να λάβουμε την τιμή μετενεργοποίησης (post-activation value) που είναι και η έξοδος του νευρώνα.



Σχήμα 3: Το μοντέλο του τεχνητού νευρώνα (Hammad, 2024)

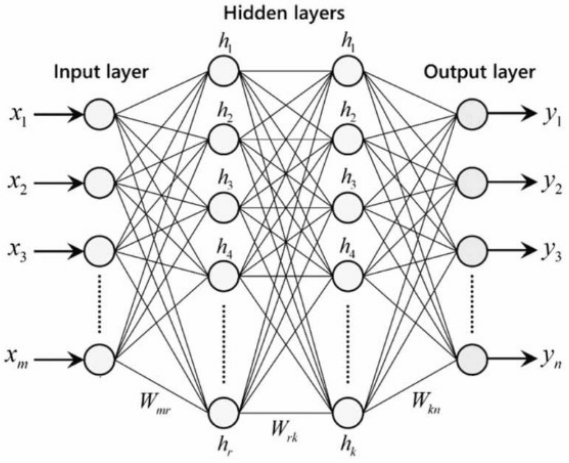
Όπως φαίνεται στο σχήμα 3, αρχικά υπολογίζεται η τιμή προ-ενεργοποίησης (pre-activation value): $z=\sum_{i=1}^{n} w_i x_i + b = \boldsymbol{w}^T \cdot \boldsymbol{x} + b$ όπου $\boldsymbol{x}=\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ το διάνυσμα των εισόδων , $\boldsymbol{w}=\begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$ το διάνυσμα των βαρών και b το bias. Κατόπιν εφαρμόζουμε την συνάρτηση ενεργοποίησης σ (AF – activation function) και παίρνουμε την βαθμωτή έξοδο $\alpha=\sigma(z)=\sigma(\sum_{i=1}^{n} w_i x_i + b)=\sigma(\boldsymbol{w}^T \cdot \boldsymbol{x} + b)$.

Η συνάρτηση ενεργοποίησης παίζει σημαντικό ρόλο στην εκπαίδευση του δικτύου και εξαρτάται από το πρόβλημα που θέλουμε να προσεγγίσουμε. Συνήθεις επιλογές είναι η σιγμοειδής (Sigmoid) ή λογιστική συνάρτηση $\sigma(z)=\frac{1}{1+e^{-z}}$ και η υπερβολική εφαπτομένη $\tanh(z)$.(Hammad, 2024)

1.4 Τεχνητά Νευρωνικά Δίκτυα

Ένα τεχνητό νευρωνικό δίκτυο (ANN) αποτελείται από ένα σύνολο τεχνητών νευρώνων και τις μεταξύ τους συνδέσεις. Η οργανωτική και λειτουργική δομή των νευρώνων και των συνδέσεών τους καθορίζουν την αρχιτεκτονική ενός Τεχνητού Νευρωνικού Δικτύου. Με βάση την ροή της πληροφορίας μπορούμε να διακρίνουμε δύο κατηγορίες δικτύων: τα δίκτυα στα οποία η πληροφορία διαδίδεται μονόδρομα από τις εισόδους προς τις εξόδους (Feed Forward Networks ή FFN) και τα δίκτυα με αναδρομικές αρχιτεκτονικές (FeedBack ή Recurrent) στα οποία η πληροφορία μπορεί να διαδίδεται αμφίδρομα ή να υπάρχουν και κυκλικές συνδέσεις.(Pires κ.ά., 2023)

Στην παρούσα εργασία θα ασχοληθούμε αποκλειστικά με Feed Forward Networks.



Σχήμα 4: Feed Forward Network (Pires κ.ά., 2023)

Στο σχήμα 4 έχουμε ένα Feed Forward δίκτυο και παρατηρούμε ότι οι νευρώνες ομαδοποιούνται σε στρώματα (layers). Μπορούμε να διακρίνουμε τα παρακάτω στρώματα:

- Input Layer: το στρώμα εισόδου όπου απλά οι νευρώνες διαμοιράζουν τις εισόδους στους νευρώνες του επόμενου στρώματος. Μερικές φορές αυτό το στρώμα παραλείπεται διότι οι νευρώνες του δεν επιτελούν κάποια λειτουργία εκτός από το να διαμοιράσουν τις εισόδους.
- Hidden Layers: τα κρυφά στρώματα όπου γίνεται η επεξεργασία των εισόδων. Εδώ οι νευρώνες υπολογίζουν το σταθμισμένο άθροισμα των εισόδων και εφαρμόζουν την συνάρτηση ενεργοποίησης για να παράγουν τις δικές τους εξόδους. Νευρωνικά δίκτυα με πάνω από ένα κρυφό στρώμα καλούνται πολυεπίπεδα (multilayer) ή βαθιά νευρωνικά δίκτυα. Αξίζει να σημειώσουμε ότι το δίκτυο του σχήματος είναι πλήρως συνδεδεμένο (fully connected) δηλαδή οι νευρώνες ενός στρώματος συνδέονται με όλους τους νευρώνες του επόμενου στρώματος.
- Output Layer: το στρώμα εξόδου όπου οι νευρώνες του παράγουν τις τελικές εξόδους του νευρωνικού δικτύου. Οι νευρώνες του στρώματος εξόδου συνήθως δεν διαφέρουν δομικά από τους νευρώνες των κρυφών στρωμάτων όμως συχνά μπορεί να διαφέρουν σε κάποια χαρακτηριστικά όπως για παράδειγμα μπορεί να έχουν διαφορετική συνάρτηση ενεργοποίησης.

  Στην πράξη μπορεί να έχουμε κάποια επιπλέον στρώματα είτε στην είσοδο είτε στην έξοδο που πραγματοποιούν συγκεκριμένους μετασχηματισμούς των δεδομένων.

1.5 Η συνάρτηση ενεργοποίησης

Η συνάρτηση ενεργοποίησης μοντελοποιεί την ενεργοποίηση ενός νευρώνα. Η κατάλληλη επιλογή της συνάρτησης ενεργοποίησης είναι θεμελιώδους σημασίας για την εκπαίδευση ενός νευρωνικού δικτύου και εξαρτάται από το είδος του προβλήματος που θέλουμε να προσεγγίσουμε.

Το 1989 οι Hornik κ.α έδειξαν ότι μια τυχαία συνεχής συνάρτηση που ορίζεται σε ένα συμπαγές υποσύνολο μπορεί να προσεγγισθεί με οποιαδήποτε ακρίβεια από ένα FFN (feed-forward network) με μόνο ένα κρυφό στρώμα (hidden layer) αρκεί να έχουμε ικανό αριθμό νευρώνων και η συνάρτηση ενεργοποίησης να είναι μια μη-σταθερή, φραγμένη και μονοτονικά αύξουσα συνεχής συνάρτηση. Το θεώρημα αυτό είναι γνωστό ως Universal Approximation Theorem και έδειξε ότι οι γραμμικές συναρτήσεις ενεργοποίησης που χρησιμοποιήθηκαν στα πρώτα νευρωνικά δίκτυα δεν μπορούν να προσεγγίσουν κάθε συνεχή συνάρτηση. Έτσι για πολλά χρόνια χρησιμοποιήθηκαν φραγμένες συναρτήσεις όπως η λογιστική ή σιγμοειδής συνάρτηση (Cybenko, 1988) και η

υπερβολική εφαπτομένη (Chen, 1990). Παρ όλα αυτά οι παραπάνω συναρτήσεις σε βαθιά δίκτυα εμφανίζουν το πρόβλημα μηδενισμού της κλίσης (vanishing gradient problem) όπως έδειξαν οι Bengio κ.ά. το 1994. Το 1993 οι Leshno, Lin, Pinkus, και Schocken έδειξαν ότι όλες οι μη – πολυωνυμικές συναρτήσεις συμπεριλαμβανομένων και των μη φραγμένων όπως η ReLU μπορούν να προσεγγίσουν με οποιαδήποτε ακρίβεια μια συνεχή συνάρτηση (δηλ. είναι universal approximators). Για την ακρίβεια το θεώρημα αναφέρεται στις μη- πολυωνυμικές *τοπικά φραγμένες* συναρτήσεις.

Συναρτήσεις όπως η ReLU δεν εμφανίζουν το πρόβλημα μηδενισμού της κλίσης όπως η σιγμοειδής και υπερβολική εφαπτομένη και έχουν καλύτερη επίδοση σε βαθιά νευρωνικά δίκτυα. Τα τελευταία χρόνια έχουν αντικαταστήσει τις φραγμένες συναρτήσεις στα βαθιά δίκτυα. Παρ όλα αυτά η ReLU εμφανίζει τα δικά της προβλήματα όπως αναφέρουμε παρακάτω και έχουν προταθεί πολλές παραλλαγές της.(Apicella κ.ά., 2020)

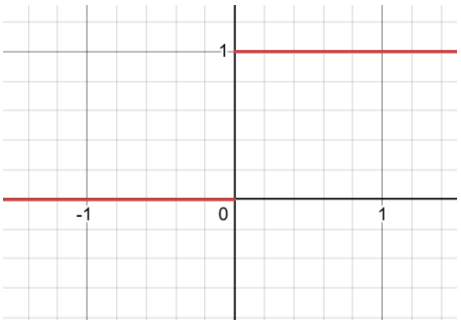
Παρακάτω αναφέρουμε μερικές κλασικές συναρτήσεις ενεργοποίησης:

- Συνάρτηση βήματος

  Ορίζεται ως $f(z)=\begin{cases}1, z\geq 0\\ 0, z<0\end{cases}$ ή $f(z)=\begin{cases}1, z\geq \theta\\ 0, z<\theta\end{cases}$,

  όπου θ το κατώφλι.

  Η συνάρτηση βήματος είναι από τις πρώτες συναρτήσεις ενεργοποίησης και χρησιμοποιήθηκε στο Perceptron που ανέπτυξε ο Frank Rosenblatt.



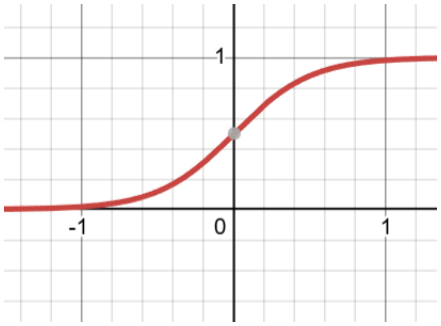
Σχήμα 5: Συνάρτηση βήματος

  Σήμερα δεν χρησιμοποιείται συχνά διότι εκτός από το κατώφλι όπου δεν ορίζεται η παράγωγος, παντού αλλού είναι μηδέν και δεν μπορεί να χρησιμοποιηθεί σε μεθόδους μάθησης όπως η Gradient descent.

- Λογιστική ή Σιγμοειδής συνάρτησης (Sigmoid)

  Ορίζεται ώς $f(z)=\sigma(z)=\frac{1}{1+e^{-az}}$

  Η Σιγμοειδής συνάρτηση μπορεί να θεωρηθεί σαν την συνεχή και παραγωγίσιμη έκδοση της συνάρτησης βήματος. Συνήθως η παράμετρος α είναι 1, αλλά καθώς



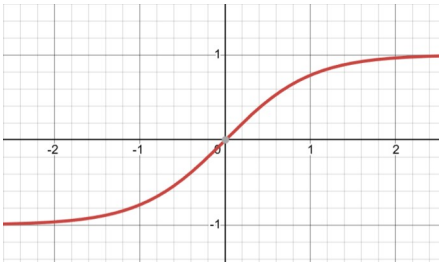
Σχήμα 6: Η Λογιστική ή Σιγμοειδής συνάρτηση

αυξάνεται γίνεται όλο και πιο απότομη και μοιάζει με την συνάρτηση βήματος. Στο σχήμα 6 α=5 για λόγους ευκρίνειας.

Αν και χρησιμοποιήθηκε συχνά στο παρελθόν εμφανίζει δύο προβλήματα. Το πρώτο πρόβλημα είναι ο κορεσμός της κλίσης (gradient saturation) για μεγάλες θετικές ή αρνητικές τιμές και το δεύτερο είναι οι πολύ μικρές τιμές της κλίσης (vanishing gradient) όταν εκπαιδεύουμε βαθιά δίκτυα με το αλγόριθμο back-propagation.

- Υπερβολική εφαπτομένη

Ορίζεται ώς $\tanh(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}=2\sigma(2z)-1$



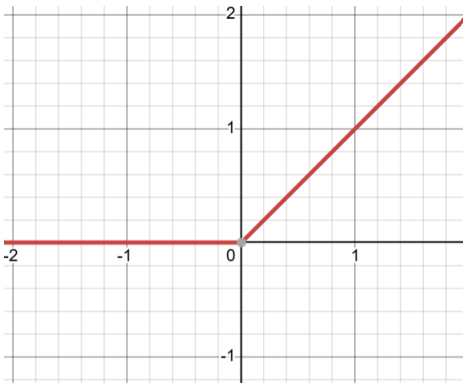
Σχήμα 7: Η υπερβολική εφαπτομένη

Η υπερβολική εφαπτομένη έχει το πλεονέκτημα ότι είναι συμμετρική και για τιμές κοντά στο μηδέν έχει κλίση μοναδιαίας συνάρτησης και έτσι η εκπαίδευση του δικτύου είναι ευκολότερη αρκεί οι έξοδοι να παραμένουν σε μικρές τιμές.

Παρ όλα αυτά εμφανίζει τα ίδια προβλήματα με την σιγμοειδή συνάρτηση.

- Rectified Linear Unit (ReLU)

Ορίζεται ως $f(z)=max(0,z)$



Σχήμα 8: Η συνάρτση ReLU

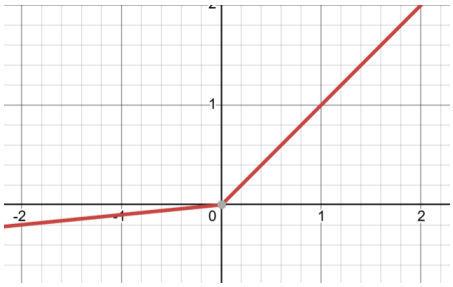
Είναι η πιο δημοφιλής συνάρτηση ενεργοποίησης λόγω της απλότητας και της καλής επίδοσης ειδικά σε βαθιά δίκτυα. Δεν έχει τα προβλήματα του μηδενισμού και του κορεσμού της κλίσης που εμφανίζουν οι σιγμοειδείς συναρτήσεις όπως η sigmoid και η tanh και οι υπολογισμοί των παραγώγων της είναι πολύ πιο απλές υπολογιστικά. Επίσης έχει αποδειχθεί ότι μπορεί να προσεγγίσει οποιαδήποτε ομαλή συνάρτηση (είναι universal approximator) . Παρ όλα αυτά έχει και μειονεκτήματα. Κατά την διάρκεια της εκπαίδευσης μπορεί κάποιοι νευρώνες να απενεργοποιηθούν μόνιμα.

Στην περίπτωση που κάποιος νευρώνας δώσει μηδενική έξοδο τα βάρη του νευρώνα σταματούν να ενημερώνονται διότι ο αλγόριθμος back propagation δίνει μηδενικές κλίσεις και σταματά η εκπαίδευση του νευρώνα.

- Leaky ReLU (LReLU)

  Ορίζεται ώς $f(z)=\begin{cases} z, z\geq 0 \\ z/\alpha, z<0 \end{cases}$

  όπου η τιμή της παραμέτρου α πρέπει να είναι "μεγάλη". Μία τυπική τιμή είναι α=100. Η Leaky ReLU λύνει το πρόβλημα των ανενεργών νευρώνων που παρουσιάζει η ReLU.



Σχήμα 9: Leaky ReLU

Ο κατάλογος των συναρτήσεων ενεργοποίησης είναι μεγάλος καθώς αποτελούν ενεργό περιοχή έρευνας στα νευρωνικά δίκτυα. (Kunc & Kléma, 2024)

Εδώ θα πρέπει να σημειωθεί ότι αν και οι ReLU και LReLU χρησιμοποιούνται ευρέως σήμερα δεν είναι κατάλληλες για τα PINNs (Physics Informed Neural Networks) διότι στα δίκτυα αυτά εισάγονται εξισώσεις φυσικής και κατά την διάρκεια της εκπαίδευσης απαιτείται υπολογισμός παραγώγων μεγαλύτερων της πρώτης τάξης. Επομένως αφού η δεύτερη παράγωγος αυτών των συναρτήσεων είναι μηδέν δεν μπορούν να χρησιμοποιηθούν για την εκπαίδευση των PINNs. (Rowan, 2025)

1.6 Η συνάρτηση σφάλματος

Σκοπός της εκπαίδευσης ενός νευρωνικού δικτύου είναι η προσαρμογή των παραμέτρων του (συνήθως των βαρών) έτσι ώστε να δίνει τις επιθυμητές εξόδους. Επομένως χρειάζεται ένα μέτρο της απόστασης των εξόδων του δικτύου από τις επιθυμητές εξόδους (true values).Την απόσταση αυτή υλοποιεί η συνάρτηση σφάλματος ή συνάρτηση απώλειας.

Στην παρούσα εργασία θα εκπαιδεύσουμε ένα νευρωνικό δίκτυο ώστε να μπορεί να προβλέψει μια συνεχή τιμή εξόδου βάσει ενός ή περισσοτέρων χαρακτηριστικών εισόδου. Ένα τέτοιο πρόβλημα καλείται πρόβλημα παλινδρόμησης (Regression). Εδώ θα πρέπει να τονίσουμε ότι τα PINNs μπορούν να εκπαιδευτούν και χωρίς εξωτερικά δεδομένα απλά θέτοντας τον περιορισμό οι έξοδοι να ικανοποιούν ένα πρόβλημα αρχικών/συνοριακών συνθηκών στο επίπεδο της συνάρτησης σφάλματος όπως θα αναφέρουμε πιο αναλυτικά στο κεφάλαιο 2.

Μία συνάρτηση σφάλματος παλινδρόμησης (regression loss function) L ορίζεται σε ένα σύνολο δεδομένων $D=\{(x_i, y_i)\}_{i=1}^{n}$ όπου $x_i \in \mathbb{R}^d$ είναι το διάνυσμα εισόδων και $y_i \in \mathbb{R}$ η συνεχής τιμή εξόδου. Έστω $\hat{y}_i = f_\theta(x_i)$ η τιμή εξόδου του δικτύου και θ το διάνυσμα παραμέτρων του δικτύου (συνήθως το διάνυσμα των βαρών). Ο σκοπός της εκπαίδευσης του δικτύου είναι να βρεθεί το διάνυσμα παραμέτρων θ που ελαχιστοποιεί την συνάρτηση σφάλματος: $\min_\theta L(\{(x_i, y_i)\}_{i=1}^{n}, \theta)$.

Η συνάρτηση σφάλματος πρέπει να διαθέτει μια σειρά από ιδιότητες που κάθε φορά λαμβάνονται υπ όψιν ανάλογα με το πρόβλημα που θέλουμε να προσεγγίσουμε :

- Κυρτότητα (Convexity):

  Μια συνάρτηση είναι κυρτή όταν κάθε τοπικό ελάχιστο αποτελεί και καθολικό ελάχιστο. Οι κυρτές συναρτήσεις σφάλματος είναι επιθυμητές, διότι δίνουν καλά αποτελέσματα με τη χρήση μεθόδων βελτιστοποίησης βασισμένων στην κλίση.

- Διαφορισιμότητα (Differentiability):

  Μια συνάρτηση σφάλματος είναι διαφορίσιμη όταν η παράγωγός της ως προς τις παραμέτρους του μοντέλου υπάρχει και είναι συνεχής. Η διαφορισιμότητα είναι απαραίτητη, διότι επιτρέπει την εφαρμογή μεθόδων βελτιστοποίησης που βασίζονται στην κλίση.

- Ανθεκτικότητα (Robustness):

  Οι συναρτήσεις σφάλματος θα πρέπει να είναι ικανές να αντιμετωπίζουν ακραίες τιμές (outliers) και να μην επηρεάζονται σημαντικά από έναν μικρό αριθμό ακραίων παρατηρήσεων.

- Ομαλότητα (Smoothness):

  Μια συνάρτηση σφάλματος θα πρέπει να διαθέτει συνεχή κλίση και να μην παρουσιάζει απότομες μεταβάσεις ή αιχμές, γεγονός που διευκολύνει τη σταθερή σύγκλιση κατά τη διαδικασία βελτιστοποίησης.

- Αραιότητα (Sparsity):

  Μια συνάρτηση σφάλματος που προάγει την αραιότητα ενθαρρύνει το μοντέλο να παράγει αραιές εξόδους. Η ιδιότητα αυτή είναι ιδιαίτερα χρήσιμη σε προβλήματα υψηλών διαστάσεων και σε περιπτώσεις όπου ο αριθμός των σημαντικών χαρακτηριστικών είναι μικρός.

- Μονοτονία (Monotonicity):

  Μια συνάρτηση σφάλματος είναι μονοτονική όταν η τιμή της μειώνεται καθώς η προβλεπόμενη έξοδος προσεγγίζει την πραγματική τιμή. Η μονοτονία διασφαλίζει ότι η διαδικασία βελτιστοποίησης κατευθύνεται προς τη σωστή λύση.

Παρακάτω αναφέρονται οι πιο συχνά χρησιμοποιούμενες συναρτήσεις σφάλματος σε προβλήματα παλινδρόμησης:

- Συνάρτηση μέσου τετραγωνικού σφάλματος MSE (Mean Squared Error)

  Είναι από τις πιο συχνά χρησιμοποιούμενες συναρτήσεις σφάλματος και αναφέρεται επίσης και ως L2 loss. Ορίζεται ως: $MSE(\theta)=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2=\frac{1}{n}\sum_{i=1}^{n}(y_i-f_\theta(x_i))^2$.

  Η συνάρτηση μέσου τετραγωνικού σφάλματος είναι θετικά ορισμένη, ομαλά διαφορίσιμη και κυρτή σε σχέση με τις εξόδους του δικτύου $\hat{y}_i$. Σχετικά με την κυρτότητα θα πρέπει να τονίσουμε ότι σε βαθιά δίκτυα με μη γραμμικές συναρτήσεις ενεργοποίησης η τελική συνάρτηση σφάλματος είναι συνήθως μη κυρτή σε σχέση με τα βάρη του δικτύου θ.

  Ένα μειονέκτημα της συνάρτησης MSE είναι η ευαισθησία στις ακραίες τιμές λόγω του τετραγώνου. Έτσι στις περιπτώσεις που έχουμε ακραίες τιμές συνήθως χρησιμοποιούνται πιο ανθεκτικές (Robust) συναρτήσεις όπως το μέσο απόλυτο σφάλμα (MAE) και η συνάρτηση Huber Loss.

- Συνάρτηση μέσου απόλυτου σφάλματος MAE (Mean Absolute Error)

  Η συνάρτηση μέσου απόλυτου σφάλματος που αναφέρεται και ως L1 loss είναι επίσης μια πολύ συχνή επιλογή. Ορίζεται ως: $MAE(\theta)=\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|=\frac{1}{n}\sum_{i=1}^{n}|y_i-f_\theta(x_i)|$.

  Η συνάρτηση MAE είναι θετικά ορισμένη , κυρτή ως προς τις εξόδους $\hat{y}_i$ και πιο ανθεκτική σε ακραίες τιμές σε σχέση με την MSE. Το κύριο πρόβλημα της MAE είναι ότι δεν είναι διαφορίσιμη στο μηδέν δηλαδή όταν $y_i=\hat{y}_i$. Το πρόβλημα αυτό αντιμετωπίζεται χρησιμοποιώντας μεθόδους γνωστές ως subgradient methods. Τυπικά αυτές οι μέθοδοι ορίζουν την παράγωγο ως προς $\hat{y}_i$ ως εξής: $\partial_{\hat{y}_i}|y_i-\hat{y}_i|=\begin{cases}1, \hat{y}_i>y_i\\ -1, \hat{y}_i<y_i\\ [-1,1], \hat{y}_i=y_i\end{cases}$.

- Η συνάρτηση Huber Loss

  Η συνάρτηση προτάθηκε από τον Huber και συνδυάζει τα πλεονεκτήματα των συναρτήσεων MSE και MAE. Έστω y η πραγματική τιμή , $\hat{y}$ η τιμή που προβλέπει το δίκτυο και δ μια παράμετρος κατωφλιού που ορίζουμε. Τότε η συνάρτηση Huber για ένα δείγμα ορίζεται ως: $L_{Huber}(y,\hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2, εάν |y-\hat{y}| \le \delta \\ \delta(|y-\hat{y}| - \frac{1}{2}\delta), αλλιώς \end{cases}$.

  Όταν το σφάλμα είναι μικρότερο από το κατώφλι η συνάρτηση συμπεριφέρεται όπως η MSE ενώ για τιμές μεγαλύτερες του κατωφλιού έχουμε την συμπεριφορά της MAE. Έτσι συνδυάζεται η ομαλότητα της MSE για μικρές τιμές και η γραμμική συμπεριφορά της MAE για μεγάλες και ακραίες τιμές. Η συνάρτηση Huber χρησιμοποιείται όταν έχουμε δεδομένα με θόρυβο και ακραίες τιμές. Το μειονέκτημά της είναι η επιπλέον υπερπαράμετρος δ που πρέπει να καθορισθεί ανάλογα με τα δεδομένα που έχουμε κάθε φορά. (Terven κ.ά., 2025)

1.7 Gradient descent και αλγόριθμοι βελτιστοποίησης

1.7.1 Ο αλγόριθμος Gradient descent

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο σκοπός της εκπαίδευσης ενός νευρωνικού δικτύου είναι να βρεθεί το διάνυσμα των παραμέτρων θ που ελαχιστοποιεί την συνάρτηση σφάλματος $L(\theta)$. Αυτή η διαδικασία ονομάζεται βελτιστοποίηση (optimization). Ο πιο συχνά χρησιμοποιούμενος αλγόριθμος βελτιστοποίησης είναι ο Gradient descent. Ο Gradient descent υπολογίζει την κλίση (Gradient) της βαθμωτής συνάρτησης σφάλματος. Το διάνυσμα της κλίσης δείχνει προς την κατεύθυνση μέγιστης αύξησης της συνάρτησης. Επειδή ο σκοπός μας είναι να βρούμε το ελάχιστο της συνάρτησης ο αλγόριθμος ενημερώνει τα βάρη προς της αντίθετη κατεύθυνση της βαθμίδας. Επίσης χρησιμοποιεί μια υπεπαράμετρο η γνωστή ως ρυθμός μάθησης (learning rate) που καθορίζει πόσο μεγάλα είναι τα βήματα με τα οποία προσεγγίζουμε ένα ελάχιστο. Επομένως ο αλγόριθμος ενημερώνει το διάνυσμα των βαρών θ σύμφωνα με τον κανόνα:

$$\theta \Leftarrow \theta - \eta \nabla_\theta L(\theta)$$

Υπάρχουν τρεις βασικές παραλλαγές του αλγόριθμου ανάλογα με το υποσύνολο των δεδομένων που χρησιμοποιούμε για την ενημέρωση των βαρών:

- Batch gradient descent

  Είναι η πιο απλή παραλλαγή του αλγορίθμου και συχνά αναφέρεται και σαν Vanilla gradient descent. Χρησιμοποιεί όλα τα δεδομένα για κάθε ενημέρωση των παραμέτρων και η υπερπαράμετρος learning rate καθορίζει πόσο μεγάλη ή μικρή θα είναι η ενημέρωση. Η σύγκλιση είναι σίγουρη σε ένα ολικό ελάχιστο για κυρτές συναρτήσεις ή σε τοπικό ελάχιστο για μη κυρτές. Λόγω των υπολογισμών σε όλο το εύρος των δεδομένων είναι αργός και δεν είναι χρήσιμος όταν το πλήθος των δεδομένων είναι μεγάλο σε σχέση με την διαθέσιμη μνήμη.

- Stochastic gradient descent (SGD)

  Σε αυτή την παραλλαγή χρησιμοποιούμε ένα ζεύγος δεδομένων εισόδου-εξόδου σε κάθε ενημέρωση των παραμέτρων. Αφού χρησιμοποιήσουμε όλα τα ζεύγη αναδιατάσσουμε τυχαία τα δεδομένα (shuffle) και επαναλαμβάνουμε. Ο αλγόριθμος SGD είναι πολύ πιο γρήγορος σε σχέση με τον batch gradient descent και κατάλληλος για μεγάλα σετ δεδομένων. Επίσης μπορεί να αποφύγει παγίδευση σε τοπικά ελάχιστα ή σαγματικά σημεία λόγω των διακυμάνσεων που παρουσιάζει. Όμως οι διακυμάνσεις που παρουσιάζει δυσκολεύουν την σύγκλιση του αλγορίθμου. Το πρόβλημα της σύγκλισης μπορεί να αντιμετωπισθεί μειώνοντας αργά τον ρυθμό μάθησης.

- Mini-batch gradient descent (MBGD)

  Ο αλγόριθμος Mini-batch gradient descent αποτελεί μια ενδιάμεση λύση και συνδυάζει τα πλεονεκτήματα των δύο προηγούμενων παραλλαγών. Για κάθε ενημέρωση των παραμέτρων χρησιμοποιεί ένα μικρό τμήμα των δεδομένων (mini-batch). Αν το τμήμα έχει n δεδομένα τότε η ενημέρωση των βαρών γίνεται σύμφωνα με τον κανόνα:

  $$\theta \Leftarrow \theta - \eta \nabla_{\theta} L\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right)$$

  Με αυτόν τον τρόπο η σύγκλιση είναι πιο ομαλή χωρίς μεγάλο υπολογιστικό κόστος. Τυπικά μεγέθη των τμημάτων είναι 50 με 256 δεδομένα άλλα συχνά το μήκος του τμήματος εξαρτάται από την εφαρμογή που έχουμε κάθε φορά. Ο αλγόριθμος Mini-batch gradient descent είναι η τυπική επιλογή για την εκπαίδευση των νευρωνικών δικτύων και ο όρος SGD συχνά αναφέρεται σε αυτόν τον αλγόριθμο. Ο κώδικας που υλοποιεί τον αλγόριθμο σε γλώσσα python δίνεται στο *παράρτημα Α*.

Παρ όλα αυτά ο αλγόριθμος mini-batch gradient descent δεν εξασφαλίζει την βέλτιστη σύγκλιση καθώς αντιμετωπίζει μια σειρά από προβλήματα. Παρακάτω αναφέρουμε τα δύο βασικά προβλήματα:

- Η επιλογή του κατάλληλου ρυθμού μάθησης είναι δύσκολη. Πολύ μικρός ρυθμός μάθησης κάνει την σύγκλιση αργή ενώ ο μεγάλος ρυθμός μάθησης μπορεί να εμποδίσει την σύγκλιση προκαλώντας ταλαντώσεις γύρω από ένα ελάχιστο ή και απόκλιση. Το πρόβλημα μπορεί να αντιμετωπισθεί με έναν μεταβλητό (συνήθως μειούμενο) ρυθμό μάθησης σύμφωνα με κάποιο μοτίβο όμως και πάλι είναι δύσκολο να προσαρμόσουμε τον ρυθμό στις ανάγκες διαφορετικών δεδομένων. Επίσης ο αλγόριθμος MBGD εφαρμόζει τον ίδιο ρυθμό μάθησης σε όλες τις παραμέτρους του δικτύου. Αν έχουμε αραιά δεδομένα ή πολλαπλές συχνότητες μπορεί να είναι επιθυμητό να έχουμε διαφορετικούς ρυθμούς μάθησης σε κάποιες παραμέτρους.
- Ο εγκλωβισμός σε ένα τοπικό ελάχιστο ή ένα σαγματικό (saddle point) σημείο. Οι συναρτήσεις σφάλματος είναι συχνά έντονα μη-κυρτές σε σχέση με τις παραμέτρους του δικτύου. Επομένως μπορεί να έχουν πολλά τοπικά ελάχιστα και σαγματικά σημεία. Ειδικά τα σαγματικά σημεία αποτελούν την μεγαλύτερη πρόκληση διότι συνήθως έχουν επίπεδα όπου η κλίση είναι σχεδόν μηδενική.

(Ruder, 2017)

Για την αντιμετώπιση των παραπάνω προβλημάτων έχουν προταθεί διάφοροι αλγόριθμοι βελτιστοποίησης. Στην παρούσα εργασία θα χρησιμοποιήσουμε κυρίως τον αλγόριθμο adam που βασίζεται στην μέθοδο της ορμής (Momentum). Η μέθοδος της ορμής δεν ταυτίζεται με την έννοια της ορμής στην Φυσική αν και η ανακάλυψή της σχετίζεται με το μηχανικό πρόβλημα μιας βαριάς σφαίρας που κατηφορίζει σε ανώμαλο τοπίο.

1.7.2 Η μέθοδος της ορμής (Momentum)

Ο Polyak ήταν ο πρώτος που συστηματικά πρότεινε την ιδέα της ορμής το 1964 εισάγοντας την μέθοδο της βαριάς σφαίρας (heavy ball method). Η ιδέα της ορμής ενσωματώθηκε σε πολλούς αλγόριθμους βελτιστοποίησης και αναδείχθηκε σε βασική τεχνική στην εκπαίδευση των νευρωνικών δικτύων. Ο Polyak χρησιμοποίησε τον παρακάτω αναδρομικό αλγόριθμο:

$$u_{t+1}=\beta u_t-\eta\nabla_\theta L(\theta_t),\quad \theta_{t+1}=\theta_t+u_{t+1},\quad \beta\in[0,1)$$

όπου η παράμετρος β είναι ο συντελεστής ορμής. Σήμερα χρησιμοποιείται μια ελαφρώς παραλλαγμένη μορφή:

$$u_{t+1}=\beta u_t+(1-\beta)\nabla_\theta L(\theta_t),\quad \theta_{t+1}=\theta_t-\eta u_{t+1},\quad \beta\in[0,1)$$

με τυπική τιμή για τον συντελεστή ορμής $\beta=0.9$.

Διαισθητικά ο αλγόριθμος του Polyak μπορεί να κατανοηθεί με μια αναλογία από την φυσική. Αν η συνάρτηση σφάλματος θεωρηθεί σαν μια συνάρτηση δυναμικής ενέργειας $U=L(\theta)$ τότε η κλίση δίνει την δύναμη που ασκείται σε ένα σώμα καθώς κινείται στην επιφάνεια της δυναμικής ενέργειας δηλ. $F=-\nabla U=-\nabla_\theta L(\theta)$. Αν θεωρήσουμε και μία δύναμη τριβής ανάλογη της ταχύτητας τότε η αναδρομική σχέση του Polyak περιγράφει την κίνηση μίας σφαίρας σε ανώμαλο έδαφος (με κοιλάδες και κορυφές) με συντελεστή τριβής β και κλίμακα χρόνου τον ρυθμό μάθησης η.

Η μέθοδος της ορμής αντικαθιστά την κλίση με έναν κινητό μέσο όρο κλίσεων. Δηλαδή για την ενημέρωση των παραμέτρων λαμβάνεται υπόψιν η τωρινή κλίση και οι προηγούμενες τιμές της κλίσης. Με αυτό τον τρόπο όταν η κλίση δεν αλλάζει σημαντικά συσσωρεύεται ορμή που βοηθά στην αποφυγή σαγματικών σημείων και τοπικών ελαχίστων ενώ όταν οι κλίσεις αλλάζουν απότομα οι συνεισφορές αλληλοαναιρούνται και έτσι έχουμε μείωση του θορύβου και τοπικών ταλαντώσεων με αποτέλεσμα γρηγορότερη σύγκλιση. Παρ όλα αυτά η μεγάλη συσσώρευση ορμής μπορεί να δημιουργήσει πρόβλημα στον εντοπισμό ελαχίστων. Έτσι χρειάζεται προσεκτική επιλογή του συντελεστή ορμής και κατάλληλη μείωση του ρυθμού μάθησης ώστε να έχουμε καλά αποτελέσματα. (Fucheng κ.ά., 2025)

1.7.3 Adam optimizer

Ο αλγόριθμος Adam (Adaptive Moment Estimation) είναι μια μέθοδος προσαρμοζόμενου ρυθμού μάθησης που τα τελευταία χρόνια χρησιμοποιείται συστηματικά για την εκπαίδευση νευρωνικών δικτύων. Βασίζεται στην ιδέα της ορμής και προσαρμόζει τον ρυθμό μάθησης κάθε παραμέτρου με βάση το ιστορικό των κλίσεων. Προτάθηκε από τους Kingma και Ba το 2014 με 2015 και συνδυάζει ορμές πρώτης και δεύτερης τάξης. Είναι ένας αλγόριθμος βελτιστοποίησης γενικού σκοπού που έχει καλές επιδόσεις σε προβλήματα με αραιά δεδομένα και θόρυβο. Σήμερα είναι ένας από τους πιο συχνά χρησιμοποιούμενους αλγόριθμους πρώτης -τάξης.

Ο Adam χρησιμοποιεί τον εκθετικά σταθμισμένο μέσο όρο των κλίσεων για να υπολογίσει την πρώτη ορμή m_t και τον εκθετικά σταθμισμένο μέσο όρο των τετραγώνων των κλίσεων (δεύτερη στατιστική ροπή) για να υπολογίσει την δεύτερη ορμή u_t:

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)g_t,\quad u_t=\beta_2 u_{t-1}+(1-\beta_2)g_t^2,\quad g_t=\nabla_\theta L(\theta_t),\quad \beta_1,\beta_2\in[0,1)$$

Επειδή $m_0=u_0=0$ ο αλγόριθμος έχει μία αρχική πόλωση προς μηδενικές κλίσεις. Για την διόρθωση της αρχικής πόλωσης εισάγεται μια διόρθωση στις ορμές:

$$\hat{m}_t=\frac{m_t}{1-\beta_1^t},\qquad \hat{u}_t=\frac{u_t}{1-\beta_2^t}.$$

Έτσι ο κανόνας διόρθωσης των βαρών είναι: $\theta_{t+1}=\theta_t-\eta\frac{\hat{m}_t}{\sqrt{\hat{u}_t}+\varepsilon}$, όπου η ο ολικός ρυθμός μάθησης.

Θα πρέπει να σημειωθεί ότι όλες οι πράξεις γίνονται ξεχωριστά για κάθε παράμετρο (element-wise). Κάθε παράμετρος έχει τον δικό της ρυθμό μάθησης ανάλογα με τις κλίσεις προς την κατεύθυνση της συγκεκριμένης συντεταγμένης. Έτσι συντεταγμένες όπου το μέτρο της κλίσης έχει μεγάλη τιμή θα έχουν μικρό ρυθμό μάθησης. Τυπικές τιμές για τις υπερπαραμέτρους είναι $\beta_1=0.9$, $\beta_2=0.999$, $\varepsilon=10^{-8}$.

(Fucheng κ.ά., 2025)

1.8 Ο αλγόριθμος Backpropagation

Για την ενημέρωση των βαρών ενός δικτύου με την μέθοδο Gradient descent απαιτείται ο υπολογισμός της κλίσης της συνάρτησης σφάλματος για κάθε ζεύγος εισόδου εξόδου του δικτύου. Κατά την εκπαίδευση απαιτούνται εκατοντάδες χιλιάδες ή και εκατομμύρια υπολογισμοί της κλίσης. Επομένως είναι αναγκαίος ένας αλγόριθμος που να υπολογίζει τις κλίσεις με αποδοτικό τρόπο. Ο αλγόριθμος που χρησιμοποιείται σήμερα είναι ο Backpropagation. *Για λόγους σαφήνειας θα συμβολίσουμε την συνάρτηση σφάλματος ως J.*

Ο αλγόριθμος backpropagation αναπτύχθηκε μεταξύ 1970 και 1980 και τελειοποιήθηκε από τους Rumelhart et al. Στόχος του αλγόριθμου είναι να υπολογίσει τις μερικές παραγώγους της συνάρτησης σφάλματος $J(W)$ σε σχέση με τα βάρη του δικτύου W . Σε κάθε κρυφό στρώμα l αντιστοιχίζεται ένας όρος σφάλματος δ^l . Για κάθε κρυφό στρώμα l ο όρος σφάλματος δ^l

υπολογίζεται από το σφάλμα δ^{l+1}, δηλαδή το σφάλμα υπολογίζεται από την έξοδο προς την είσοδο γιαυτό και ο αλγόριθμος ονομάζεται backpropagation.

Το σφάλμα δ^L του στρώματος εξόδου L δεν έχει εξαρτήσεις από άλλα σφάλματα και σε μορφή πινάκων δίνεται από την σχέση:

$\delta^L=\frac{\partial J}{\partial \alpha^L}\odot\sigma'(z^L)=\nabla_{\alpha^L}J\odot\sigma'(z^L)$, όπου $\alpha^l=\sigma(z^l)$ οι έξοδοι του στρώματος l , $z^l=W^l\alpha^{l-1}+b^l$ η τιμή προ ενεργοποίησης του στρώματος l και W^l ο πίνακας βαρών. Το σύμβολο $\odot$ δηλώνει τον πολλαπλασιασμό δύο πινάκων ίσων διαστάσεων στοιχείο επί στοιχείο (Hadamard product).

Κατόπιν τα σφάλματα των υπόλοιπων στρωμάτων υπολογίζονται από την αναδρομική σχέση:

$$\delta^l=((W^{l+1})^T\delta^{l+1})\odot\sigma'(z^l),\quad l=(L-1),....,1$$

Όταν το σφάλμα υπολογισθεί για κάποιο στρώμα τότε η κλίση της συνάρτησης σφάλματος ως προς τα βάρη μπορεί να βρεθεί από την σχέση:

$$\frac{\partial J}{\partial W^l}=\delta^l(\alpha^{l-1})^T$$

(Boughammoura, 2023)

Ο αλγόριθμος backpropagation χρησιμοποιεί τον κανόνα της αλυσίδας (chain rule) ώστε να υπολογίσει τις μερικές παραγώγους. Όμως αντί να υπολογίσει κατευθείαν την επίδραση των βαρών στην συνάρτηση σφάλματος υπολογίζει την επίδραση της τιμής προ ενεργοποίησης κάθε κόμβου. Με αυτόν τον τρόπο οι υπολογισμοί είναι πιο γρήγοροι και είναι πιο εύκολο να καταλήξουμε σε μια αναδρομική σχέση. Επίσης αυτός ο τρόπος είναι πιο φυσικός αφού το σφάλμα σε κάθε κόμβο οφείλεται στα σφάλματα των προηγούμενων κόμβων μέσω της τιμής προ ενεργοποίησης z_i^l.

Για την εξαγωγή του αλγόριθμου θα χρησιμοποιήσουμε συνιστώσες με τον ακόλουθο συμβολισμό:

- w_{ij}^l το j - βάρος του i - νευρώνα που βρίσκεται στο l - στρώμα. Δηλαδή το βάρος της σύναψης μεταξύ του j νευρώνα στο $l-1$ στρώμα με τον i - νευρώνα στο l στρώμα.
- b_i^l η πόλωση του i - νευρώνα που βρίσκεται στο l - στρώμα.
- α_i^l η έξοδος του i - νευρώνα που βρίσκεται στο l - στρώμα.
- z_i^l η τιμή προ ενεργοποίησης του i - νευρώνα που βρίσκεται στο l - στρώμα.

Επομένως ισχύει η σχέση : $\alpha_i^l=\sigma(z_i^l)=\sigma(\sum_j w_{ij}^l \alpha_j^{l-1}+b_i^l)$. Επίσης υποθέτουμε ότι έχουμε k νευρώνες ανά στρώμα.

Θέλουμε να υπολογίσουμε τις ποσότητες $\frac{\partial J}{\partial w_{ik}^l}$. Από το κανόνα της αλυσίδας έχουμε:

$$\frac{\partial J}{\partial w_{ik}^l}=\frac{\partial J}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{ik}^l}=\frac{\partial J}{\partial z_i^l}\alpha_k^{l-1}.$$

Θέτουμε $\delta_i^l=\frac{\partial J}{\partial z_i^l}$ επομένως $\frac{\partial J}{\partial w_{ik}^l}=\delta_i^l\alpha_k^{l-1}$ όπου δ_i^l το σφάλμα στον νευρώνα i του l στρώματος. Άρα θα πρέπει να βρούμε μια αναδρομική σχέση για τον υπολογισμό των σφαλμάτων δ_i^l. Αρχικά θα υπολογίσουμε το σφάλμα για την έξοδο στο στρώμα $l=L$ και κατόπιν θα βρούμε μια αναδρομική σχέση που συνδέει το σφάλμα στο στρώμα l με το σφάλμα στο στρώμα $l+1$.

- Σφάλμα στην έξοδο: $\delta_i^L=\frac{\partial J}{\partial z_i^L}=\frac{\partial J}{\partial \alpha_i^L}\frac{\partial \alpha_i^L}{\partial z_i^L}=\frac{\partial J}{\partial \alpha_i^L}\sigma'(z_i^L)$ όπου η ποσότητα $\frac{\partial J}{\partial \alpha_i^L}$ είναι εύκολο να υπολογισθεί αφού εξαρτάται μόνο από την συνάρτηση σφάλματος. Για παράδειγμα αν $J=\frac{1}{2}\sum_i(\alpha_i^L-y_i)^2$ τότε $\frac{\partial J}{\partial \alpha_i^L}=(\alpha_i^L-y_i)$ και σε μορφή πινάκων $\delta^L=\nabla_a J\odot\sigma'(z^L)=(\alpha^L-y)\odot\sigma'(z^L)$.

- Για το σφάλμα $\delta_i^l=\frac{\partial J}{\partial z_i^l}$ σε ένα ενδιάμεσο κρυφό στρώμα έχουμε: $\delta_i^l=\frac{\partial J}{\partial z_i^l}=\sum_k\frac{\partial J}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_i^l}=\sum_k\delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_i^l}$ και ο τελευταίος όρος δίνεται από την σχέση:

  $$\frac{\partial z_k^{l+1}}{\partial z_i^l}=\frac{\partial}{\partial z_i^l}\sum_j(w_{kj}^{l+1}\alpha_j^l+b_k^{l+1})=\frac{\partial}{\partial z_i^l}\sum_j(w_{kj}^{l+1}\sigma(z_j^l)+b_k^{l+1})=w_{ki}^{l+1}\sigma'(z_i^l).$$

  Άρα η αναδρομική σχέση είναι: $\delta_i^l=\sum_k\delta_k^{l+1}w_{ki}^{l+1}\sigma'(z_i^l)$ ή σε μορφή πινάκων $\delta^l=((W^{l+1})^T\delta^{l+1})\odot\sigma'(z^l)$.

Τέλος για τον υπολογισμό των πολώσεων χρησιμοποιούμε την σχέση $\frac{\partial J}{\partial b_i^l}=\delta_i^l$.

Όπως φαίνεται από τις παραπάνω σχέσεις για τον υπολογισμό των σφαλμάτων πρέπει πρώτα να υπολογισθούν οι τιμές προ ενεργοποίησης z_i^l και ενεργοποίησης α_i^l των νευρώνων. Άρα ο αλγόριθμος ξεκινά από την είσοδο προς την έξοδο και υπολογίζει όλες τις εξόδους των νευρώνων

(Forward pass). Κατόπιν κινείται αντίστροφα από την έξοδο προς τις εισόδους υπολογίζοντας τα σφάλματα (backward pass). Η έξοδος του αλγόριθμου είναι οι κλίσεις της συνάρτησης σφάλματος. Οι παράγωγοι που απαιτούνται για τον υπολογισμό του σφάλματος υπολογίζονται με μια τεχνική που ονομάζεται αυτόματη διαφόριση (automatic differentiation)

(*GEO5017 - Machine Learning for the Built Environment*, χ.χ.; Nielsen, 2015)

1.9 Αυτόματη διαφόριση (Automatic differentiation)

Η αυτόματη διαφόριση (automatic differentiation ή AD) είναι η μέθοδος που χρησιμοποιείται σήμερα για τον υπολογισμό των μερικών παραγώγων που απαιτούνται κατά την εκπαίδευση των νευρωνικών δικτύων. Συγκεκριμένα ο αλγόριθμος backpropagation υλοποιείται μέσω μιας μορφής της αυτόματης διαφόρισης που ονομάζεται αντίστροφη μορφή (reverse mode AD). Η αυτόματη διαφόριση έχει δύο βασικές μορφές Forward mode και Reverse mode. Η ευθεία μορφή (Forward mode AD) σαν μια γενική μέθοδος για τον υπολογισμό μερικών παραγώγων ανακαλύφθηκε από τον Wengert το 1964. Ο Linnainmaa (1970, 1976) θεωρείται ο πρώτος που δημοσίευσε μια περιγραφή της αντίστροφης μορφής (reverse mode). Ο Speelpenning (1980) εισήγαγε το reverse mode AD στην μορφή που έχει σήμερα.

Η AD δεν είναι αριθμητική παραγώγιση αν και υπολογίζει αριθμητικές τιμές παραγώγων. Η AD υπολογίζει τις παραγώγους χωρίς σφάλματα στο όριο της ακρίβειας του υπολογιστικού συστήματος. Επίσης δεν είναι συμβολική παραγώγιση παρόλο που χρησιμοποιεί κανόνες συμβολικής παραγώγισης διότι αποθηκεύει ενδιάμεσες αριθμητικές τιμές και δεν παράγει τελικές εκφράσεις.

Η αριθμητική παραγώγιση χρησιμοποιεί εξισώσεις πεπερασμένων διαφορών που είναι απλές στην υλοποίηση αλλά έχουν προβλήματα λόγω των σφαλμάτων στρογγυλοποίησης και απαιτείται προσεκτική επιλογή του βήματος διακριτοποίησης. Το κύριο πρόβλημα είναι ότι υπολογίζει την κλίση σε n διαστάσεις με πολυπλοκότητα $O(n)$. Στην περίπτωση των νευρωνικών δικτύων όπου έχουμε εκατομμύρια ή και δισεκατομμύρια διαστάσεις αυτό αποτελεί σημαντικό εμπόδιο για την χρήση της μεθόδου.

Η συμβολική διαφόριση είναι ο αυτόματος χειρισμός εκφράσεων ώστε να πάρουμε εκφράσεις παραγώγων. Εφαρμόζει μετασχηματισμούς που βασίζονται στους κανόνες διαφόρισης. Υλοποιείται σε σύγχρονα συστήματα υπολογιστικής άλγεβρας όπως τα Mathematika , Maxima , Maple και σε

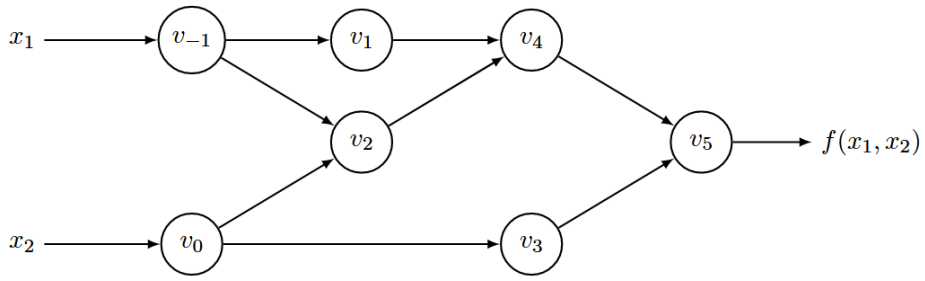
frameworks μηχανικής μάθησης όπως η Theano. Η συμβολική διαφόριση μπορεί να παράγει εκθετικά μεγάλες εκφράσεις αν δεν χρησιμοποιηθεί με προσοχή. Η AD έχει το πλεονέκτημα ότι δεν χρειάζεται να παράγει τελικές εκφράσεις και έτσι αποθηκεύει ενδιάμεσες αριθμητικές τιμές. Εφαρμόζει συμβολική διαφόριση μόνο σε ενδιάμεσες απλές εκφράσεις. Αυτό μπορεί να μειώσει σημαντικά τους απαιτούμενους υπολογισμούς.

Επομένως η αυτόματη διαφόριση δίνει ακριβή αριθμητικά αποτελέσματα με μικρό υπολογιστικό κόστος. Αυτό την καθιστά ιδανική μέθοδο για τον υπολογισμό των κλίσεων σε βαθιά νευρωνικά δίκτυα.

Η αυτόματη διαφόριση βασίζεται στο ότι όλοι οι αριθμητικοί υπολογισμοί μπορούν να αναλυθούν σε ένα πεπερασμένο σύνολο στοιχειωδών υπολογισμών όπου οι παράγωγοι είναι γνωστοί εκ των προτέρων. Και στις δύο μορφές της AD αρχικά έχουμε μια διαδικασία όπου ξεκινώντας από τις εισόδους υπολογίζουμε όλες τις ενδιάμεσες τιμές καθώς και τις εξόδους. Ταυτόχρονα δημιουργείται το υπολογιστικό γράφημα που χρησιμοποιείται στην δεύτερη φάση για τον υπολογισμό μερικών παραγώγων με τον κανόνα της αλυσίδας. Αυτή η πρώτη φάση ονομάζεται forward pass.

Για να δούμε στην πράξη πως λειτουργεί η AD θα χρησιμοποιήσουμε ένα παράδειγμα.

Έστω ότι θέλουμε να υπολογίσουμε την μερική παράγωγο $\frac{\partial y}{\partial x_1}$ της συνάρτησης $y=f(x_1,x_2)=\ln(x_1)+x_1x_2-\sin(x_2)$ στο σημείο $(x_1,x_2)=(2,5)$. Αρχικά έχουμε το υπολογιστικό γράφημα όπου φαίνονται οι ενδιάμεσες μεταβλητές και οι εξαρτήσεις τους.



Σχήμα 10 Υπολογιστικό γράφημα (Baydin κ.ά., 2018)

Κατόπιν θα δείξουμε τα ίχνη των υπολογισμών για τις δύο μορφές της AD.

- Forward Mode AD

Για να υπολογίσουμε την μερική παράγωγο $\frac{\partial y}{\partial x_1}$ ξεκινάμε αντιστοιχίζοντας σε κάθε ενδιάμεση μεταβλητή u_i την παράγωγο $\dot{u}_i = \frac{\partial u_i}{\partial x_1}$. Κατόπιν υπολογίζουμε την μερική παράγωγο κάθε ενδιάμεσης μεταβλητής εφαρμόζοντας τον κανόνα της αλυσίδας. Οι υπολογισμοί έχουν κατεύθυνση από τις μεταβλητές εισόδου προς τις μεταβλητές εξόδου όπως φαίνεται στο παρακάτω σχήμα.

Forward Primal Trace			Forward Tangent (Derivative) Trace		
v_{-1}	$= x_1$	$= 2$	$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
v_0	$= x_2$	$= 5$	$\dot{v}_0$	$= \dot{x}_2$	$= 0$
v_1	$= \ln v_{-1}$	$= \ln 2$	$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$	$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
v_3	$= \sin v_0$	$= \sin 5$	$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$	$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$	$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
y	$= v_5$	$= 11.652$	$\dot{y}$	$= \dot{v}_5$	$= 5.5$

Σχήμα 11 Forward mode AD (Baydin κ.ά., 2018)

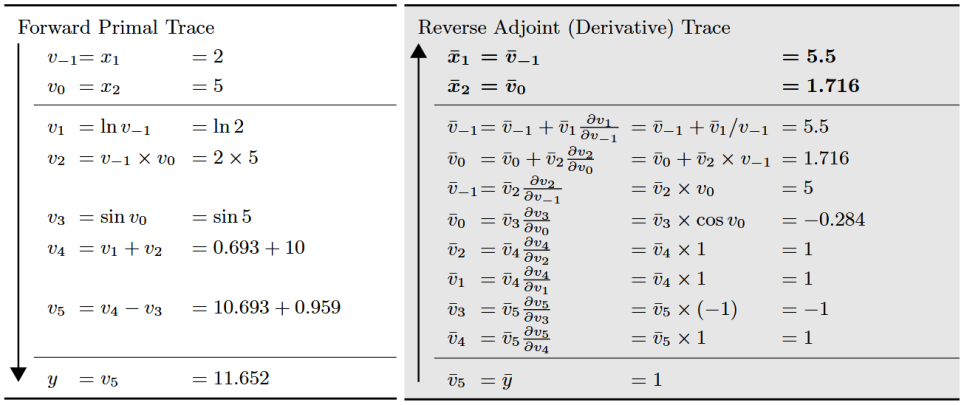
Η ευθεία μορφή είναι ιδιαίτερα αποτελεσματική σε συναρτήσεις της μορφής $f : \mathbb{R} \to \mathbb{R}^n$ αφού μπορεί να υπολογίσει όλες τις παραγώγους $\frac{d y_i}{dx}$ με ένα μόνο πέρασμα.

- Reverse mode AD

Η αντίστροφη μορφή είναι μια γενίκευση του αλγόριθμου backpropagation αφού υπολογίζει τις παραγώγους από τις εξόδους προς τις εισόδους. Αυτό το πετυχαίνει συμπληρώνοντας κάθε ενδιάμεση μεταβλητή u_i με μία συζυγή (adjoint) $\bar{u}_i = \frac{\partial y}{\partial u_i}$.

Είναι χρήσιμο να δούμε την y σαν συνάρτηση σφάλματος ενός νευρωνικού δικτύου και τις ενδιάμεσες μεταβλητές u_i σαν τις τιμές προ ενεργοποίησης των νευρώνων. Τότε οι συζυγείς μεταβλητές $\bar{u}_i$ είναι τα σφάλματα που συναντήσαμε στον αλγόριθμο backpropagation.

Και εδώ χρησιμοποιώντας κατάλληλα τον κανόνα της αλυσίδας υπολογίζουμε τις συζυγείς μεταβλητές αλλά με κατεύθυνση από την έξοδο προς τις εισόδους (αντίστροφα) όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 12 Reverse mode AD (Baydin κ.ά., 2018)

Αξίζει να σημειωθεί ότι υπολογίσθηκαν όλες οι παράγωγοι της εξόδου ως προς τις εισόδους δηλαδή $\frac{\partial y}{\partial x_1}$ και $\frac{\partial y}{\partial x_2}$ με ένα μόνο reverse pass.

Επομένως σε μια συνάρτηση της μορφής $f : \mathbb{R}^n \to \mathbb{R}$ *ο reverse mode AD έχει πολυπλοκότητα* $O(1)$ *και επομένως εξαιρετική απόδοση σε σχέση με άλλους αλγόριθμους.*

Εν γένει για μία συνάρτηση $f : \mathbb{R}^n \to \mathbb{R}^m$ ο αλγόριθμος reverse mode AD έχει καλύτερη απόδοση όταν $m \ll n$. Πάντως αν και αποδοτικός από άποψη υπολογισμών ο reverse AD χρειάζεται αυξημένη μνήμη που στην χειρότερη περίπτωση είναι ανάλογη του αριθμού των πράξεων. Η βελτίωση των απαιτήσεων σε μνήμη αποτελεί ενεργό τομέα έρευνας.

(Baydin κ.ά., 2018)

2 Physics-Informed Neural Networks (PINNs)

Στην φυσική , την μηχανική αλλά και σε πολλές άλλες επιστήμες τα διάφορα φαινόμενα συχνά μοντελοποιούνται με την μορφή διαφορικών εξισώσεων (Δ.Ε). Οι διαφορικές εξισώσεις είναι απαραίτητες για την ερμηνεία της συμπεριφοράς των συστημάτων και εξαρτώνται από μια ή περισσότερες μεταβλητές και τους ρυθμούς μεταβολής τους. Για απλά φυσικά φαινόμενα ή συστήματα οι Δ.Ε μπορούν να λυθούν και να πάρουμε αναλυτικές λύσεις. Όμως στις περισσότερες περιπτώσεις και ειδικά σε περιπτώσεις με πρακτικό ενδιαφέρον δεν μπορούμε να έχουμε αναλυτικές λύσεις. Έτσι έχουν αναπτυχθεί διάφορες αριθμητικές μέθοδοι όπως η μέθοδος των πεπερασμένων στοιχείων , η μέθοδος των πεπερασμένων διαφορών κ.α. Με την πάροδο του χρόνου έχουν βρεθεί μαθηματικά εργαλεία που εξασφαλίζουν την ακρίβεια, την ευστάθεια και την σύγκλιση αυτών των μεθόδων. Όμως οι αριθμητικές μέθοδοι παρουσιάζουν δυσκολίες όσον αφορά την αποδοτικότητά τους όταν αντιμετωπίζουμε πολύπλοκα προβλήματα ή και προβλήματα με μεγάλο αριθμό διαστάσεων. Σε αυτές τις τεχνικές η ακρίβεια και η υπολογιστική απόδοση επηρεάζονται σε μεγάλο βαθμό από την διακριτοποίηση και την δημιουργία του κατάλληλου πλέγματος. Ειδικά σε ακανόνιστες γεωμετρίες , προβλήματα μη επαρκώς τοποθετημένα ή με ελλιπείς συνθήκες και προβλήματα με πολλές διαστάσεις οι δυσκολίες είναι μεγάλες.

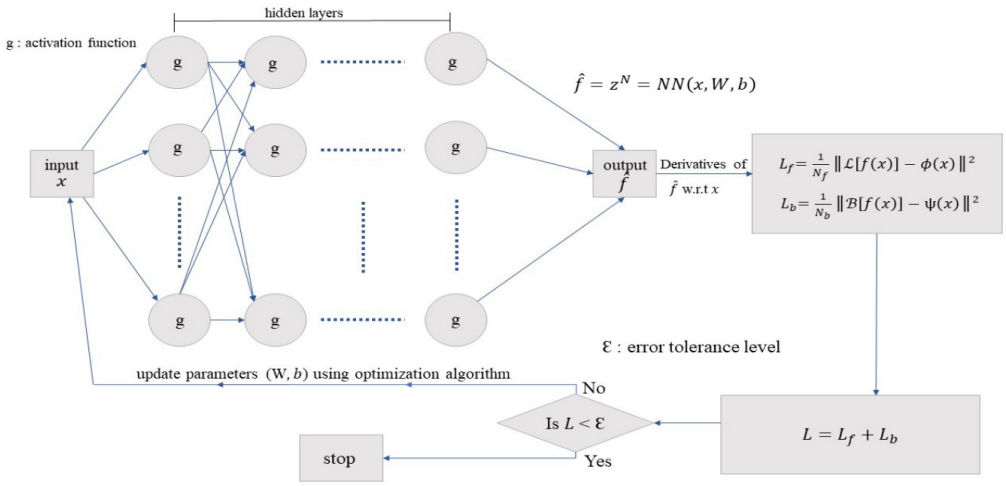
Λόγω των παραπάνω δυσκολιών οι ερευνητές δείχνουν όλο και μεγαλύτερο ενδιαφέρον για τα PINNs (Physics Informed Neural Networks). Τα PINNs μετατρέπουν το πρόβλημα της εύρεσης μιας προσεγγιστικής λύσης σε πρόβλημα ελαχιστοποίησης μιας συνάρτησης σφάλματος με την χρήση νευρωνικών δικτύων. Συγκεκριμένα εισάγουν στην συνάρτηση σφάλματος του νευρωνικού δικτύου το άθροισμα των υπολοίπων (Residuals) των αρχικών και συνοριακών συνθηκών καθώς και των υπολοίπων της διαφορικής εξίσωσης, σε επιλεγμένα σημεία εντός του χωρίου λύσης (collocation points). Στην ουσία ενσωματώνεται η φυσική του προβλήματος στο πλαίσιο (framework) των νευρωνικών δικτύων. Εδώ θα πρέπει να τονίσουμε ότι τα PINNs μπορούν να λειτουργήσουν και χωρίς εξωτερικά δεδομένα. Δηλαδή μπορούν να επιλύσουν μια Δ.Ε απλά εισάγοντας την μορφή του διαφορικού τελεστή και τις απαραίτητες αρχικές και συνοριακές συνθήκες σε μια συνάρτηση σφάλματος. Παρ όλα αυτά είναι αποτελεσματικά σε προβλήματα με λίγα δεδομένα και σε αντίστροφα προβλήματα (inverse problems) όπου από πειραματικά δεδομένα θέλουμε να προσδιορίσουμε κάποιες παραμέτρους.

Τα PINNs δεν χρειάζονται την δημιουργία πλέγματος και έχουν καλές επιδώσεις σε προβλήματα μεγάλων διαστάσεων. Επίσης μπορούν χειριστούν με επιτυχία προβλήματα με πολύπλοκες γεωμετρίες.

Παρά τα καινοτόμα χαρακτηριστικά τους παρουσιάζουν ορισμένους εγγενείς περιορισμούς. Αντιμετωπίζουν τα προβλήματα του μηδενισμού της κλίσης (vanishing gradient) που συναντώνται σε βαθιά δίκτυα. Η εκπαίδευση των PINNs συχνά απαιτεί περισσότερο χρόνο. Λόγω της πρόσφατης ανάπτυξής τους στερούνται αποτελεσματικών εργαλείων ανάλυσης σφάλματος. Η κατανόηση των κριτηρίων σύγκλισης των PINNs αποτελεί πρόκληση και είναι ενεργό πεδίο έρευνας. Τέλος οι πρόσθετοι όροι στην συνάρτηση σφάλματος μπορούν να δημιουργήσουν πρόβλημα κατά την διαδικασία βελτιστοποίησης διότι ενδέχεται κάποιοι όροι να έχουν σημαντικά μεγαλύτερες κλίσεις από τους υπόλοιπους και να δημιουργήσουν προκατάληψη (bias).

Μια από τις πρώτες εργασίες για την επίλυση διαφορικών εξισώσεων με νευρωνικά δίκτυα είναι των I. Lagaris, A. Likas, D. Fotiadis (1998). Στην αρχική τους εργασία η λύση μιας Δ.Ε προσεγγίζονταν από μία δοκιμαστική συνάρτηση (test function) με δύο όρους. Ο πρώτος ικανοποιούσε τις συνοριακές/αρχικές συνθήκες και ο δεύτερος προσεγγίζονταν από ένα νευρωνικό δίκτυο. Ο όρος physics-informed neural networks (PINNs) εισήχθη για πρώτη φορά το 2017 με τις εργασίες των Raissi et al. Οι συγγραφείς παρουσίασαν την λύση ευθέων και αντίστροφων προβλημάτων καθώς και αλγόριθμους συνεχούς και διακριτού χρόνου.

Στο παρακάτω σχήμα δίνεται μια αναπαράσταση ενός PINN:



*Σχήμα 13 Σχηματική αναπαράσταση ενός PINN (*Ganga & Uddin, 2024)

Το PINN του σχήματος 13 προσεγγίζει την λύση της μερικής διαφορικής εξίσωσης (PDE) $L[f(x)]=\varphi(x) \quad \forall x\in\Omega$ όπου L ο διαφορικός τελεστής και $B[f(x)]=\psi(x) \quad \forall x\in\partial\Omega$ οι συνοριακές συνθήκες. Η συνάρτηση σφάλματος (loss function) έχει δύο όρους που είναι το L^2 μέτρο (L^2 norm) των υπολοίπων της PDE και των συνοριακών συνθηκών.

$$L=L_f+L_b, \quad L_f=\frac{1}{N_f}\|L[f(x)]-\varphi(x)\|^2 \quad x\in\Omega_f, \quad L_b=\frac{1}{N_b}\|B[f(x)]-\psi(x)\|^2 \quad x\in\Omega_b$$

όπου Ω_f, Ω_b τα σύνολα των επιλεγμένων σημείων (collocation points) από τα σύνολα Ω και $\partial\Omega$ αντίστοιχα. Επίσης συνήθως οι όροι της συνάρτησης σφάλματος πολλαπλασιάζονται με κατάλληλα βάρη.

Για την εκπαίδευση του δικτύου χρησιμοποιούνται αλγόριθμοι τύπου backpropagation καθώς και η αυτόματη διαφόριση για τον υπολογισμό των παραγώγων. Συνήθεις αλγόριθμοι βελτιστοποίησης είναι οι Adam και L-BFGS (αλγόριθμος δεύτερης τάξης).

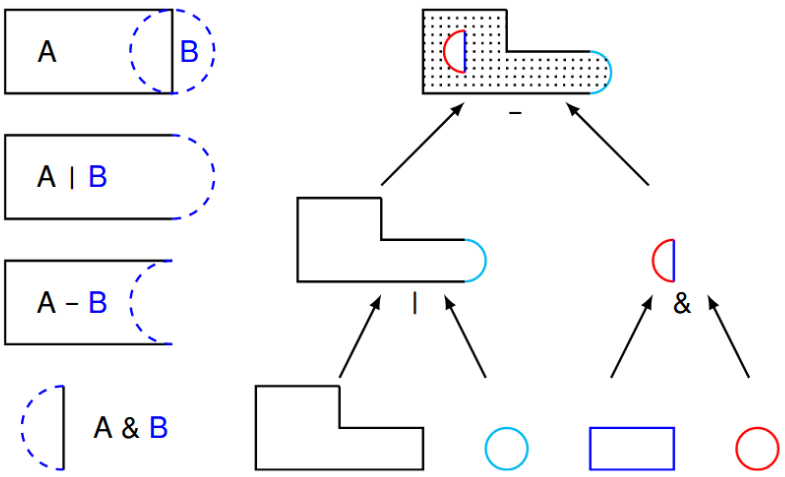
Τέλος θα πρέπει να τονίσουμε ότι στα PINNs απαιτείται ο υπολογισμός παραγώγων των εξόδων του δικτύου σε σχέση με τις εισόδους του δηλαδή παραγώγων της μορφής $\frac{\partial f(x_i)}{\partial x_i}$ και ανώτερης τάξης ανάλογα με την διαφορική εξίσωση που έχουμε κάθε φορά.

Αυτές οι παράγωγοι περιπλέκουν το τοπίο της συνάρτησης σφάλματος και κάνουν την σύγκληση των PINNs πιο περίπλοκη σε σχέση με τα απλά νευρωνικά δίκτυα. (Ganga & Uddin, 2024)

3 Η βιβλιοθήκη DeepXDE

Η DeepXDE είναι μια βιβλιοθήκη σε γλώσσα Python και είναι σχεδιασμένη ώστε να μπορεί να χρησιμοποιηθεί τόσο σαν εκπαιδευτικό εργαλείο όσο σαν ερευνητικό εργαλείο για προβλήματα υπολογιστικής επιστήμης και μηχανικής. Μπορεί να χρησιμοποιηθεί σε πολύπλοκα ρεαλιστικά προβλήματα φυσικής και υποστηρίζει σύνθετες γεωμετρίες μέσω μιας τεχνικής κατασκευής στερεάς γεωμετρίας (constructive solid geometry- CSG). Χρησιμοποιώντας την DeepXDE μπορούμε να λύσουμε χρονοεξαρτημένες μερικές διαφορικές εξισώσεις (time-PDEs) με τον ίδιο βαθμό δυσκολίας που θα λύναμε μια χρονοανεξάρτητη. Οι χρήστες μπορούν να παρακολουθήσουν και να τροποποιήσουν την διαδικασία εκπαίδευσης μέσω **callback functions**. Για παράδειγμα παρακολουθώντας το φάσμα Fourier της λύσης του νευρωνικού δικτύου μπορεί να φανεί ο τρόπος που το δίκτυο μαθαίνει συγκεκριμένες συχνότητες. Τέλος η βιβλιοθήκη είναι σχεδιασμένη ώστε ο κώδικας που γράφουμε να είναι συμπαγής και εύκολα διαχειρίσιμος προσεγγίζοντας στενά με την μαθηματική διατύπωση του προβλήματος.

Στην βιβλιοθήκη είναι ενσωματωμένες οι βασικές γεωμετρίες όπως **interval**, **triangle**, **rectangle**, **polygon**, **disk**, **cuboid** και **sphere.** Πιο σύνθετες γεωμετρίες μπορούν να κατασκευασθούν από τις βασικές χρησιμοποιώντας πράξεις αντίστοιχες με τις γνωστές πράξεις των συνόλων όπως **union (|) , difference (-)** και **intersection (&)**. Στο παρακάτω σχήμα υπάρχουν μερικά παραδείγματα κατασκευής σύνθετων γεωμετριών:

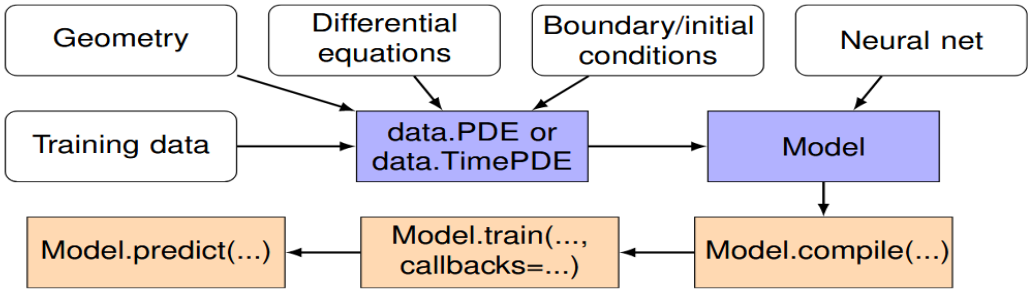


Σχήμα 14 Σύνθετες γεωμετρίες με την DeepXDE (Lu κ.ά., 2021)

Για να λύσουμε ένα πρόβλημα με την DeepXDE απαιτούνται (τουλάχιστον) τα παρακάτω βήματα:

1. Καθορίζουμε το χωρίο επίλυσης της διαφορικής εξίσωσης χρησιμοποιώντας το module **geometry**.
2. Ορίζουμε την PDE με σύνταξη παρόμοια με της βιβλιοθήκης **TensorFlow**.
3. Καθορίζουμε τις αρχικές και συνοριακές συνθήκες.
4. Συνδυάζουμε την γεωμετρία, την PDE και τις αρχικές/συνοριακές συνθήκες σε ένα αντικείμενο τύπου **`data.PDE`** για χρονοανεξάρτητα προβλήματα ή σε **`data.TimePDE`** για χρονοεξαρτημένα. Για να ορίσουμε τα σημεία εκπαίδευσης (collocation points) μπορούμε είτε να ορίσουμε συγκεκριμένα σημεία ή απλά να ορίσουμε το πλήθος των σημείων και η DeepXDE να επιλέξει τα σημεία τυχαία.
5. Κατασκευάζουμε ένα νευρωνικό δίκτυο χρησιμοποιώντας το module **`maps`**.
6. Ορίζουμε ένα **`Model`** συνδυάζοντας το αντικείμενο του βήματος 4 και το νευρωνικό δίκτυο του βήματος 5.
7. Καλούμε την **`Model.compile`** και θέτουμε τις υπερπαραμέτρους όπως ο optimizer (π.χ Adam) και τον ρυθμό μάθησης. Επίσης μπορούμε να ορίσουμε βάρη στους όρους της συνάρτησης σφάλματος χρησιμοποιώντας την παράμετρο **`loss_weights`**.
8. Καλούμε την **`Model.train`** για να εκπαιδεύσουμε το δίκτυο ξεκινώντας από μια τυχαία διαμόρφωση των βαρών ή χρησιμοποιούμε ένα προεκπαιδευμένο δίκτυο με το όρισμα **`model_restore_path`**. Εδώ είναι χρήσιμο να ορίσουμε **callback functions** για να παρακολουθήσουμε ή να τροποποιήσει την εκπαίδευση του δικτύου.
9. Καλούμε την **`Model.predict`** για να πάρουμε την προσέγγιση της λύσης σε διάφορα σημεία.

Η διαδικασία που περιγράψαμε φαίνεται και στο παρακάτω σχήμα:



Σχήμα 15 Διάγραμμα διαδικασίας επίλυσης PDE με την DeepXDE (Lu κ.ά., 2021)

Η βιβλιοθήκη υποστηρίζει τέσσερις βασικές συνοριακές συνθήκες Dirichlet (**`DirichletBC`**), Neumann (**`NeumannBC`**), Robin (**`RobinBC`**), και περιοδικές (**`PeriodicBC`**). Πιο γενικές συνοριακές συνθήκες μπορούν να ορισθούν χρησιμοποιώντας την **`OperatorBC`** ενώ οι αρχικές συνθήκες δηλώνονται με την **`IC`**. Επίσης στην τωρινή έκδοση η DeepXDE υποστηρίζει δύο τύπους δικτύων: feed-forward (**`maps.FNN`**) και residual (**`maps.ResNet`**).

Τέλος η βιβλιοθήκη μπορεί να τροποποιηθεί. Για παράδειγμα μπορούμε να δημιουργήσουμε νέες γεωμετρίες , αρχιτεκτονικές δικτύων και callback functions.

(Lu κ.ά., 2021)

4 Πειράματα με την DeepXDE

Στην παρούσα εργασία θα μελετήσουμε δύο προβλήματα της ουράνιας μηχανικής. Το πρόβλημα των δύο σωμάτων (ή πλανητών) που έχει αναλυτική λύση και το χαοτικό πρόβλημα των τριών σωμάτων. Θα χρησιμοποιήσουμε τα PINNs για να βρούμε προσεγγίσεις σε διάφορα προβλήματα αρχικών συνθηκών και θα συγκρίνουμε τις λύσεις των PINNs με τις αντίστοιχες αριθμητικές λύσεις. Για την υλοποίηση των PINNs θα χρησιμοποιήσουμε την βιβλιοθήκη **DeepXDE** και για τις αριθμητικές λύσεις την βιβλιοθήκη **Scipy** της γλώσσας python.

Και τα δυο προβλήματα μπορούν να περιγραφούν από ένα σύστημα απλών διαφορικών εξισώσεων (ODEs) δεύτερης τάξης. Συνήθως το πρόβλημα μετατρέπεται σε ODEs πρώτης τάξης για την αριθμητική επίλυση. Παρ όλα αυτά το σύστημα εξισώσεων πρώτης τάξης κάνει πολύπλοκη την συνάρτηση σφάλματος λόγω των πολλών όρων και δυσκολεύει την εκπαίδευση του νευρωνικού δικτύου. Για αυτόν το λόγο θα χρησιμοποιήσουμε και τις δύο μεθόδους και θα συγκρίνουμε τα αποτελέσματα. Αρχικά θα εκπαιδεύσουμε τα PINNs χωρίς εξωτερικά δεδομένα. Κατόπιν θα χρησιμοποιήσουμε εξωτερικά δεδομένα και θα συγκρίνουμε την απόδοση των PINNs με απλά νευρωνικά δίκτυα.

Τα προβλήματα της ουράνιας μηχανικής εμφανίζουν δύο βασικές δυσκολίες όσον αφορά την εκπαίδευση ενός νευρωνικού δικτύου. Λόγω των πολλών εξισώσεων και αρχικών συνθηκών η συνάρτηση σφάλματος έχει πολλούς όρους με διαφορετική συμπεριφορά στην εκπαίδευση. Αυτό καθιστά την εκπαίδευση δυσκολότερη και υπάρχει περίπτωση το δίκτυο να ικανοποιήσει την φυσική χωρίς να επιβάλλει τις σωστές αρχικές συνθήκες ή και το αντίστροφο. Το δεύτερο πρόβλημα είναι ο νόμος του αντιστρόφου τετραγώνου (νόμος παγκόσμιας έλξης). Όταν τα σώματα πλησιάζουν οι επιταχύνσεις τους αυξάνονται απότομα. Αυτό είναι πιθανόν το μεγαλύτερο πρόβλημα ειδικά για το χαοτικό πρόβλημα των τριών σωμάτων.

Επομένως τα προβλήματα των δύο και τριών σωμάτων απαιτούν προσεκτική ρύθμιση των υπερπαραμέτρων του δικτύου (π.χ συντελεστές βαρών στην συνάρτηση σφάλματος), κατάλληλο optimizer και επιβολή των αρχικών συνθηκών με κατάλληλους μετασχηματισμούς (hard constrains).

Τα πειράματα έγιναν με την *έκδοση 1.5 της βιβλιοθήκης deepxde* με την χρήση ενός οικιακού υπολογιστή. Στα πειράματα που έγινε χρήση εξειδικευμένου hardware από το *google colab* αυτό αναφέρεται ρητά. Επίσης σε όλα τα πειράματα κρατήσαμε σταθερό seed ίσο με 137 για τις μηχανές

παραγωγής ψευδοτυχαίων αριθμών ώστε να έχουμε επαναληψιμότητα και εύκολη σύγκριση των αποτελεσμάτων. Τέλος στα στα περισσότερα πειράματα χρησιμοποιήσαμε 64 σημεία εκπαίδευσης (collocation points) και 100 σημεία για έλεγχο της σύγκλισης (test points). Σε περιπτώσεις που χρησιμοποιήθηκε διαφορετικός αριθμός σημείων ή seed αυτό αναφέρεται ρητά.

4.1 Το πρόβλημα των δύο σωμάτων

4.1.1 Περιγραφή του προβλήματος

Το πρόβλημα των δύο σωμάτων της ουράνιας μηχανικής είναι ο υπολογισμός των τροχιών δύο σημειακών (ή σφαιρικά συμμετρικών) σωμάτων που αλληλεπιδρούν με την δύναμη της βαρύτητας. Το πρόβλημα επιλύθηκε από τον Isaak Newton στο έργο του Principia που εκδόθηκε το 1687. Το πρόβλημα μπορεί να αναχθεί σε ένα ζεύγος προβλημάτων ενός σώματος και να λυθεί αναλυτικά. Το πρώτο πρόβλημα είναι η κίνηση του κέντρου μάζας και το δεύτερο η σχετική κίνηση των δύο σωμάτων. (Goldstein κ.ά., 2002)

Αξίζει να σημειωθεί ότι η επίλυση του ρεαλιστικού προβλήματος όπου έχουμε μη σφαιρικά σώματα είναι πολύ πιο δύσκολη και συνήθως δεν έχουμε αναλυτικές λύσεις. Τα σώματα που έχουν διαστάσεις περιστρέφονται και αναπτύσσονται ροπές που εξαρτώνται από τον σχετικό προσανατολισμό των σωμάτων. (Luo, 2020)

Στην παρούσα εργασία δεν θα χρησιμοποιήσουμε τις μαθηματικές τεχνικές ώστε να απλοποιήσουμε το πρόβλημα των δύο σημειακών σωμάτων διότι θέλουμε να χρησιμεύσει σαν εισαγωγικό πρόβλημα για την επίλυση του προβλήματος των τριών σωμάτων με χρήση νευρωνικών δικτύων. Επομένως θα εισάγουμε το πρόβλημα σαν έναν σύστημα απλών διαφορικών εξισώσεων με τις αντίστοιχες αρχικές συνθήκες ώστε να παρατηρήσουμε τις δυσκολίες στην εκπαίδευση των PINNs.

4.1.2 Μαθηματική διατύπωση και διαστατική ανάλυση

Η κίνηση των δύο σωμάτων μπορεί να δειχθεί ότι περιορίζεται σε ένα επίπεδο λόγω του νόμου της διατήρησης της στροφορμής. Επομένως θα διατυπώσουμε το πρόβλημα στις δύο διαστάσεις.

Έτσι έχουμε το σύστημα απλών διαφορικών εξισώσεων δεύτερης τάξης:

$a_{xi}=\ddot{x}_i=G m_j\frac{(x_j-x_i)}{r_{12}^{\ 3}}$, $a_{yi}=\ddot{y}_i=G m_j\frac{(y_j-y_i)}{r_{12}^{\ 3}}$, $r_{12}=r_{21}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$, $i,j=1,2$ με τις αντίστοιχες αρχικές συνθήκες $x_i(0)=x_{i0}, y_i(0)=y_{i0}, \dot{x}_i(0)=v_{x_{i0}}, \dot{y}_i(0)=v_{y_{i0}}$, $i=1,2$. Το σύστημα αποτελείται από τέσσερις διαφορικές εξισώσεις και οκτώ αρχικές συνθήκες. (Goldstein κ.ά., 2002)

Μπορούμε να μετατρέψουμε το σύστημα σε Δ.Ε πρώτης τάξης με το κόστος επιπλέον τεσσάρων εξισώσεων: $\dot{x}_i = v_{xi},\ a_{xi} = \dot{v}_{xi} = G m_j \frac{(x_j - x_i)}{r_{12}^3},\ \dot{y}_i = v_{yi},\ a_{yi} = \dot{v}_{yi} = -G m_j \frac{(y_j - y_i)}{r_{12}^3}\ \ i, j = 1,2.$ Σε αυτήν την περίπτωση θα έχουμε οκτώ Δ.Ε με οκτώ αρχικές συνθήκες.

Το επόμενο βήμα είναι να μετατρέψουμε το πρόβλημα σε αδιάστατο. Πρακτικά αυτό σημαίνει να θέσουμε G=1 στην σταθερά παγκόσμιας έλξης και να θεωρήσουμε κατάλληλες κλίμακες για τα μήκη και τις μάζες.

Η εξίσωση που δίνει τις επιταχύνσεις έχει την μορφή $a = G\frac{m}{r^2}$. Αν συμβολίσουμε με L,M,T τις διαστάσεις μήκους, μάζας και χρόνου έχουμε: $[a] = [G]\frac{[m]}{[r]^2} \Leftrightarrow \frac{L}{T^2} = [G]\frac{M}{L^2} \Leftrightarrow T = \sqrt{\frac{L^3}{[G]M}}$. Επομένως επιλέγοντας κλίμακα αποστάσεων την r_0 , κλίμακα για τις μάζες την m_0, η κλίμακα χρόνου θα είναι $t_0 = \sqrt{\frac{r_0^3}{G m_0}}$. Εισάγουμε τις αδιάστατες μεταβλητές $\tau = \frac{t}{t_0}, \rho = \frac{r}{r_0}, \mu = \frac{m}{m_0}$ και με χρήση του κανόνα της αλυσίδας $\frac{d}{dt} = \frac{d}{d\tau}\frac{d\tau}{dt}$ στις εξισώσεις δεύτερης τάξης έχουμε τις αδιάστατες εξισώσεις: $\ddot{\hat{x}}_i = \mu_j \frac{(\hat{x}_j - \hat{x}_i)}{\rho_{12}^3},\ \ddot{\hat{y}}_i = \mu_j \frac{(\hat{y}_j - \hat{y}_i)}{\rho_{12}^3},\ \rho_{12} = \rho_{21} = \sqrt{(\hat{x}_1 - \hat{x}_2)^2 + (\hat{y}_1 - \hat{y}_2)^2}\ ,\ i, j = 1,2$ όπου τώρα οι χρονικές παράγωγοι είναι ως προς την αδιάστατη μεταβλητή τ και $\hat{x}, \rho$ οι αδιάστατες συντεταγμένες , αποστάσεις. Εκτός από τις εξισώσεις η αλλαγή κλίμακας θα επηρεάσει και τις αρχικές συνθήκες που καθορίζουν την μορφή της λύσης.

Επομένως θέτοντας G=1 στην σταθερά παγκόσμιας έλξης έχουμε ένα αδιάστατο πρόβλημα με την κατανόηση ότι έχουμε επιλέξει κλίμακες για τις μάζες και τις αποστάσεις που καθορίζουν μέσω της σχέσης $t_0 = \sqrt{\frac{r_0^3}{G m_0}}$ την κλίμακα χρόνου του προβλήματος. Μία τυπική επιλογή για την κλίμακα μάζας στο πρόβλημα των δύο σωμάτων είναι το άθροισμα των μαζών $m_0 = m_1 + m_2$ και για την κλίμακα μήκους μια απόσταση της τάξης της αρχικής απόστασης των σωμάτων. Τότε η κλίμακα χρόνου είναι της ίδιας τάξης μεγέθους με την περίοδο του συστήματος στην περίπτωση που έχουμε κλειστές ελλειπτικές τροχιές. Για σύγκριση αναφέρουμε ότι σύμφωνα με τον τρίτο νόμο του Kepler

η περίοδος των ελλειπτικών τροχιών είναι : $T_{period}=2\pi\sqrt{\frac{\alpha^3}{G(m_1+m_2)}}$, όπου α ο μεγάλος ημιάξονας της ελλειπτικής τροχιάς. (Goldstein κ.ά., 2002)

Μια άλλη συχνή επιλογή για την κλίμακα μάζας είναι η μάζα του μεγαλύτερου σώματος. Αυτή η επιλογή συναντάται συχνά στο πρόβλημα των τριών σωμάτων.

Κατά την εκπαίδευση των νευρωνικών δικτύων είναι συχνά επιθυμητό η κλίμακα χρόνου να έχει μικρές τιμές ώστε να βρισκόμαστε στην γραμμική περιοχή της συνάρτησης σφάλματος. Για την περίπτωση της υπερβολικής εφαπτομένης η γραμμική περιοχή είναι περίπου στο διάστημα $[-1,1]$. (Theodosiou & Rekatsinas, 2026)

Ένας τρόπος είναι να κάνουμε μια αλλαγή κλίμακας σε ένα ήδη αδιάστατο πρόβλημα χωρίς να αλλάξουμε την λύση του προβλήματος είναι ο εξής: έστω ότι θέλουμε να αυξήσουμε την κλίμακα χρόνου κατά λ ώστε να έχουμε μικρότερες τιμές για την περίοδο κίνησης. Τότε θέτοντας $\lambda=\kappa^3$ και χρησιμοποιώντας την σχέση $t_0=\sqrt{\frac{r_0^3}{G m_0}}$ έχουμε τις αλλαγές $t_0\to\frac{t_0}{\kappa^3}$, $r_0\to\frac{r_0}{\kappa^2}$, $u_0\to\kappa u_0$, όπου $u_0=\frac{r_0}{t_0}$ η κλίμακα ταχυτήτων. Με τον κανόνα της αλυσίδας μπορούμε να επιβεβαιώσουμε ότι η παραπάνω ταυτόχρονη αλλαγή στις κλίμακες χρόνου και χώρου αφήνει αναλλοίωτη την εξίσωση του συστήματος και επομένως τα συμπεράσματα που παίρνουμε μελετώντας το νέο πρόβλημα ισχύουν και για το αρχικό.

Εδώ θα πρέπει να αναφέρουμε ότι αυτή η πρακτική δεν είναι συνηθισμένη στα pinns όπου συχνά γίνονται αλλαγές κλίμακας σε διάφορα μεγέθη με κριτήριο την εύκολη εκπαίδευση του δικτύου. Σε αυτές τις περιπτώσεις θα πρέπει να θέσουμε και κατάλληλους συντελεστές στις αρχικές εξισώσεις. Τέλος αφού το δίκτυο εκπαιδευτεί θα πρέπει να επαναφέρουμε τις κλίμακες των μεγεθών στις αρχικές τους ώστε να πάρουμε την σωστή απεικόνιση των λύσεων.

Στην περίπτωση των pinns που εξετάζουμε η κλίμακα χρόνου έχει τον κυρίαρχο ρόλο στην εκπαίδευση γιαυτό συχνά χρησιμοποιούμε την αλλαγή κλίμακας που αφήνει την εξίσωση αναλλοίωτη ώστε να μην χρειάζεται επαναφορά στην αρχική κλίμακα μετά την εκπαίδευση.

4.1.3 Vanilla Pinn – Σύστημα Δ.Ε πρώτης τάξης

Το πρώτο απλό pinn (vanilla pinn) χρησιμοποιεί το σύστημα των εξισώσεων 1ης τάξης και επομένως η συνάρτηση σφάλματος έχει οκτώ όρους L_{PDE} για τα υπόλοιπα (residues) των διαφορικών εξισώσεων και οκτώ όρους L_{ICs} για τα υπόλοιπα των αρχικών συνθηκών. Το δίκτυο έχει μία είσοδο με τις χρονικές στιγμές και οκτώ εξόδους για τις συνιστώσες της θέσης και της ταχύτητας κάθε σώματος δηλ. $y=[x_1, y_1, v_{x1}, v_{y1}, x_2, y_2, v_{x2}, v_{y2}]$.

Σαν πρότυπο χρησιμοποιήθηκε το παράδειγμα A simple ODE system από την τεκμηρίωση της βιβλιοθήκης DeepXDE. Το pinn από την αρχή δίνει NAN τιμές στα υπόλοιπα των όρων με τις επιταχύνσεις και τερματίζει. Αυτό πιθανόν σημαίνει διαίρεση με μηδέν στους όρους των επιταχύνσεων διότι αρχικά το δίκτυο τοποθετεί τα δυο σώματα πολύ κοντά. Για να το διαπιστώσουμε χρησιμοποιήθηκε μια callback function που υπολογίζει στην αρχή της πρώτης εποχής εκπαίδευσης όλες τις τιμές εξόδου του δικτύου. Ο κώδικας της callback function δίνεται στο *παράρτημα Β*.

Η έξοδος του script φαίνεται στο παρακάτω σχήμα:

```
Step      Train loss
                                                Test loss
                                                                                                                                                Test m
etric
0         [1.28e-02, 1.55e-01, nan, nan, 8.73e-04, 5.40e-03, nan, nan, 4.00e+00, 4.00e+00, 4.00e+00, 4.00e+00, 4.0
0e+00, 4.00e+00, 4.00e+00, 4.00e+00]                        [1.29e-02, 1.55e-01, 2.70e+03, 2.07e+05, 8.08e-04, 5.3
9e-03, 2.65e+07, 2.03e+09, 4.00e+00, 4.00e+00, 4.00e+00, 4.00e+00, 4.00e+00, 4.00e+00, 4.00e+00, 4.00e+00]    []
[iter 0] Full PINN state at t=0.0:
Body 1: x1=0, y1=0, vx1=0, vy1=0
Body 2: x2=0, y2=0, vx2=0, vy2=0
Inter-body distance r12 = 0

Best model at step 0:
  train loss: inf
  test loss: inf
  test metric:
```

Σχήμα 16: vanilla pinn- nan values

Από την έξοδο επιβεβαιώνεται ότι το πρόβλημα οφείλεται στις αρχικές μηδενικές τιμές των εξόδων και επειδή έχουμε ορίσει τις επιταχύνσεις ως $a_{xi}=\dot{v}_{xi}=G\,m_j\dfrac{(x_j-x_i)}{r_{12}^{3}}$ έχουμε απροσδιοριστία 0/0 που δικαιολογεί την ένδειξη **nan** (not a number) στους αντίστοιχους όρους.

Ο ποιο απλός τρόπος να αντιμετωπίσουμε το πρόβλημα είναι να προσθέσουμε έναν μικρό αριθμό ε στον παρονομαστή ώστε να αποφύγουμε το πρόβλημα χωρίς να αλλάξουμε σημαντικά την φυσική του προβλήματος. Επομένως η εξίσωση για τις επιταχύνσεις θα είναι : $a_{xi}=\dot{v}_{xi}=G\,m_j\dfrac{(x_j-x_i)}{(r_{12}+\varepsilon)^3}$.

Με αυτήν την τροποποίηση λύνεται το πρόβλημα του πρόωρου τερματισμού όμως τώρα έχουμε το πρόβλημα των πολύ μεγάλων διαφοροποιήσεων στα υπόλοιπα. Για παράδειγμα τα υπόλοιπα των όρων που περιέχουν επιταχύνσεις είναι πολλές τάξεις μεγέθους μεγαλύτερα από τα υπόλοιπα των ταχυτήτων και των αρχικών συνθηκών. Το πρόβλημα μπορεί να εξομαλυνθεί σε κάποιο βαθμό με κατάλληλα βάρη όμως παρόλα αυτά το απλό pinn δεν καταφέρνει να ικανοποιήσει τις αρχικές συνθήκες του προβλήματος με αποτέλεσμα μη αποδεκτά αποτελέσματα ακόμα και μετά από μεγάλο αριθμό εποχών εκπαίδευσης.

Για παράδειγμα αναφέρουμε το ακόλουθο σενάριο: σώματα με σχετικές μάζες $m_1=0.99, m_2=0.01$ με αρχικές συνθήκες $x_1=-0.1, y_1=0.0, v_{x1}=v_{x2}=0.0, x_2=0.1, y_2=0.0, v_{x2}=0.0, v_{y2}=-2.0$ και παραμέτρους εκπαίδευσης : $\varepsilon=10^{-6}$, δίκτυο με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα με αρχικοποίηση Glorot uniform , συνάρτηση ενεργοποίησης tanh , adam optimizer με ρυθμό μάθησης 0.0001 και βάρη 1 για τα υπόλοιπα των Δ.Ε και 100 για τα υπόλοιπα των αρχικών συνθηκών μετά από 100000 εποχές δίνει τα παρακάτω αποτελέσματα.

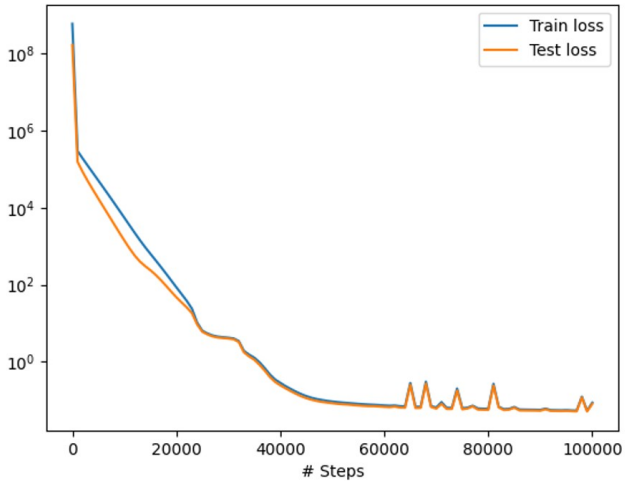
```
Step      Train loss
                                                            Test loss
                                                                                                    Test m
etric
0         [6.87e-03, 5.16e-02, 3.36e+03, 5.76e+04, 4.75e-02, 1.66e-01, 3.29e+07, 5.64e+08, 1.00e+00, 0.00e+00, 0.0
0e+00, 0.00e+00, 1.00e+00, 0.00e+00, 0.00e+00, 4.00e+02]    [6.80e-03, 5.13e-02, 9.50e+02, 1.63e+04, 4.69e-02, 1.6
6e-01, 9.32e+06, 1.60e+08, 1.00e+00, 0.00e+00, 0.00e+00, 0.00e+00, 1.00e+00, 0.00e+00, 0.00e+00, 4.00e+02]    []
1000      [8.00e-02, 1.71e-01, 2.49e+01, 3.45e+00, 1.31e-02, 5.46e-01, 2.62e+05, 3.57e+04, 7.29e-01, 1.85e-03, 7.6
5e-04, 1.70e-03, 8.08e-01, 2.46e-04, 4.95e-08, 3.62e+02]    [8.01e-02, 1.71e-01, 1.17e+01, 2.98e+00, 1.05e-02, 5.4
8e-01, 1.24e+05, 3.22e+04, 7.29e-01, 1.85e-03, 7.65e-04, 1.70e-03, 8.08e-01, 2.46e-04, 4.95e-08, 3.62e+02]    []
2000      [2.33e-01, 2.19e-01, 1.52e+01, 1.53e+00, 1.01e-01, 1.01e+00, 1.72e+05, 1.66e+04, 7.12e-01, 1.83e-03, 2.1
7e-03, 2.17e-03, 7.88e-01, 2.14e-04, 5.50e-04, 3.26e+02]    [2.33e-01, 2.20e-01, 6.27e+00, 1.09e+00, 9.46e-02, 1.0
1e+00, 7.02e+04, 1.39e+04, 7.12e-01, 1.83e-03, 2.17e-03, 2.17e-03, 7.88e-01, 2.14e-04, 5.50e-04, 3.26e+02]    []
3000      [3.73e-01, 2.14e-01, 9.65e+00, 7.65e-01, 3.23e-01, 1.61e+00, 1.13e+05, 8.25e+03, 6.90e-01, 1.85e-03, 3.4
9e-03, 2.14e-03, 7.65e-01, 1.91e-04, 2.21e-03, 2.92e+02]    [3.73e-01, 2.15e-01, 3.94e+00, 5.10e-01, 3.10e-01, 1.6
1e+00, 4.16e+04, 6.56e+03, 6.90e-01, 1.85e-03, 3.49e-03, 2.14e-03, 7.65e-01, 1.91e-04, 2.21e-03, 2.92e+02]    []
```

Σχήμα 17: vanilla-pinn: Υπόλοιπα στις πρώτες εποχές εκπαίδευσης

Αρχικά παρατηρούμε τις πολύ μεγάλες αποκλίσεις στα υπόλοιπα που κυμαίνονται από 10^{-3} έως 10^{8}. Μετά από 3000 εποχές εκπαίδευσης τα υπόλοιπα κυμαίνονται από 10^{-4} έως 10^{5}. Τα πρώτα δύο υπόλοιπα αφορούν τους όρους ταχυτήτων $\frac{dx_i}{dt}-v_{xi}$ για το βαρύ σώμα και είναι τάξης 10^{-1} ,τα υπόλοιπα 3 και 4 αφορούν τους όρους επιταχύνσεων $\frac{dv_{xi}}{dt}-a_{xi}$ για το βαρύ σώμα και είναι τάξης 1, τα υπόλοιπα 5 και 6 αφορούν τους όρους ταχυτήτων $\frac{dx_i}{dt}-v_{xi}$ για το σώμα μικρής μάζας και είναι τάξης 1 ,τα υπόλοιπα 7 και 8 αφορούν τους όρους επιταχύνσεων $\frac{dv_{xi}}{dt}-a_{xi}$ για το σώμα μικρής μάζας και είναι τάξης 10^{5} και 10^{4}. Τέλος τα υπόλοιπα 9 έως 16 αφορούν τις αρχικές συνθήκες και και κυμαίνονται από 10^{2} έως 10^{-4}.

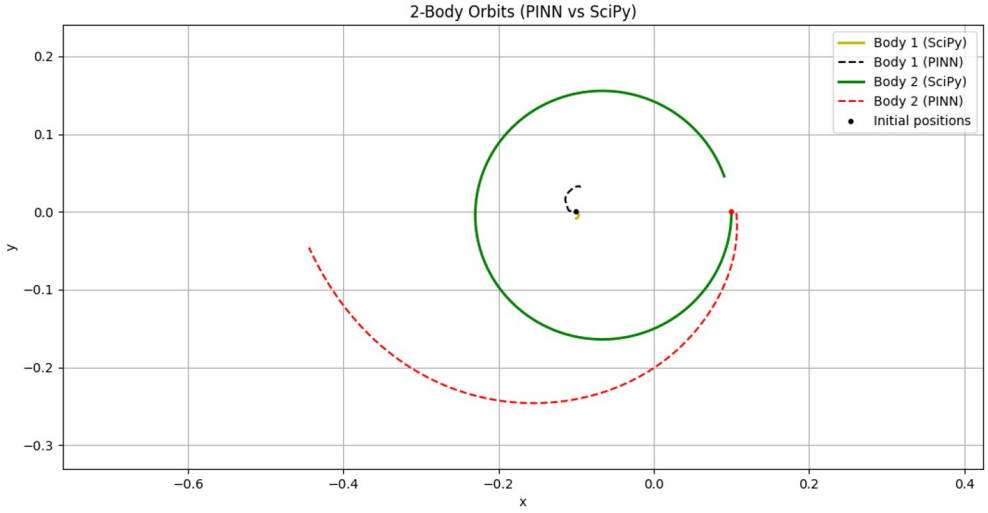
Ο αλγόριθμος gradient descent θα επιχειρήσει να μειώσει τα μεγάλα υπόλοιπα των επιταχύνσεων του μικρού σώματος και θα δώσει μικρή προτεραιότητα στο βαρύ σώμα και κυρίως στις αρχικές συνθήκες. Ο adam optimizer που χρησιμοποιούμε παρ όλο που εφαρμόζει διαφορετικούς ρυθμούς μάθησης για κάθε παράμετρο του δικτύου ανάλογα με την κλίση δεν φαίνεται να μπορεί να ξεπεράσει το πρόβλημα των πολλών υπολοίπων με τις διαφορετικές τάξεις μεγέθους.

Στα παρακάτω σχήματα φαίνονται τα αποτελέσματα της εκπαίδευσης μετά από 100000 εποχές.

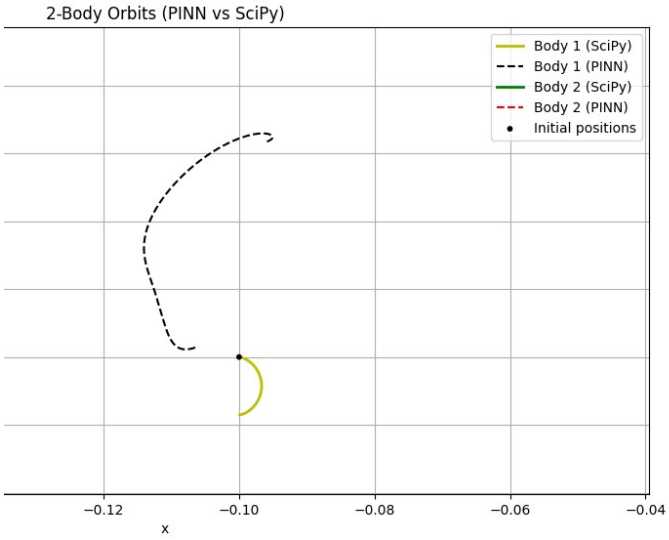


Σχήμα 18: vanilla-pinn: Διάγραμμα συνολικών απωλειών κατα την εκπαίδευση

Από το διάγραμμα απωλειών φαίνεται ότι μετά από 60000 εποχές εκπαίδευσης δεν έχουμε σημαντική μείωση των απωλειών. Η τελική τιμή είναι $5.39 \cdot 10^{-2}$.



Σχήμα 19: vanilla - pinn: Τροχιές των δύο σωμάτων



Σχήμα 20: vanilla-pinn: Λεπτομέρεια τροχιάς σώματος μεγάλης μάζας

Στο σχήμα 19 φαίνονται οι τροχιές των δύο σωμάτων ενώ στο σχήμα 20 έχουμε μεγεθύνει την τροχιά του σώματος μεγάλης μάζας ώστε να φαίνεται με μεγαλύτερη ευκρίνεια. Η λύση του Pinn είναι με διακεκομμένες γραμμές και συγκρίνεται με την αντίστοιχη αριθμητική λύση του scipy intergator. Παρατηρούμε ότι παρά τις πολλές εποχές εκπαίδευσης οι αρχικές συνθήκες δεν ικανοποιούνται αν και έχουν 100 φορές μεγαλύτερα βάρη από τα υπόλοιπα της Δ.Ε. Επίσης παρατηρούμε ότι η τροχιά του σώματος μικρής μάζας (σώμα 2) αν και δεν είναι ικανοποιητική έχει σωστή κλίση. Αντίθετα η τροχιά του σώματος μεγάλης μάζας (σώμα 1) που φαίνεται καλύτερα στο σχήμα 20 έχει αντίθετη κλίση από την αντίστοιχη αριθμητική λύση. Αυτό επιβεβαιώνει την παρατήρηση ότι λόγω των μεγάλων υπολοίπων που παρατηρούμε από την αρχή της εκπαίδευσης στο σώμα μικρής μάζας ο αλγόριθμος gradient descent θα προσπαθήσει να μειώσει το σφάλμα στο σώμα 2 δίνοντας μικρότερη προτεραιότητα στην τροχιά του άλλου σώματος.

4.1.4 Σύστημα Δ.Ε πρώτης τάξης με επιβολή των αρχικών συνθηκών (hard constrains)

Το πρόβλημα της ικανοποίησης των αρχικών συνθηκών στα Pinns είναι γνωστό ειδικά στην περίπτωση των δύσκαμπτων Δ.Ε (stiff odes-pdes). Η ικανοποίηση των αρχικών-συνοριακών συνθηκών σε συνδυασμό με την εξισορρόπηση πολλών όρων συχνά οδηγεί σε προβλήματα σύγκλισης και μη βέλτιστες λύσεις. (Hao κ.ά., 2024)

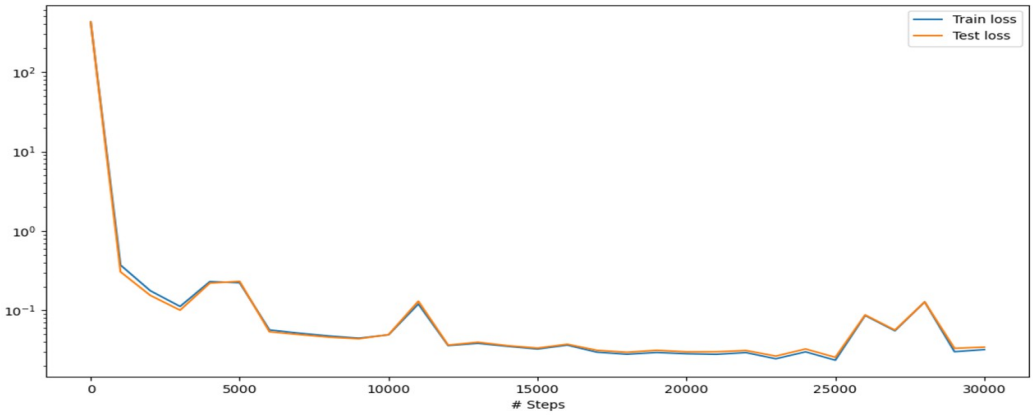
Ο όρος *stiff ODEs* προέρχεται από την αριθμητική ανάλυση και αφορά την αριθμητική επίλυση διαφορικών εξισώσεων με πολλαπλές χρονικές κλίμακες, οι οποίες διαφέρουν κατά τάξεις μεγέθους. Πρακτικά, αυτό σημαίνει ότι ορισμένες συνιστώσες της λύσης μεταβάλλονται πολύ ταχύτερα από άλλες, γεγονός που επιβάλλει τη χρήση ιδιαίτερα μικρού χρονικού βήματος διακριτοποίησης για λόγους αριθμητικής σταθερότητας. (Lambert, 1991)

Το πρόβλημα των δύο σωμάτων δεν μπορεί να χαρακτηρισθεί ως δύσκαμπτο (stiff) αλλά όπως δείξαμε στην προηγούμενη παράγραφο αντιμετωπίζουμε το πρόβλημα της ικανοποίησης των αρχικών συνθηκών και ταυτόχρονα την εξισορρόπηση πολλών όρων διαφορετικής τάξης μεγέθους στα υπόλοιπα.

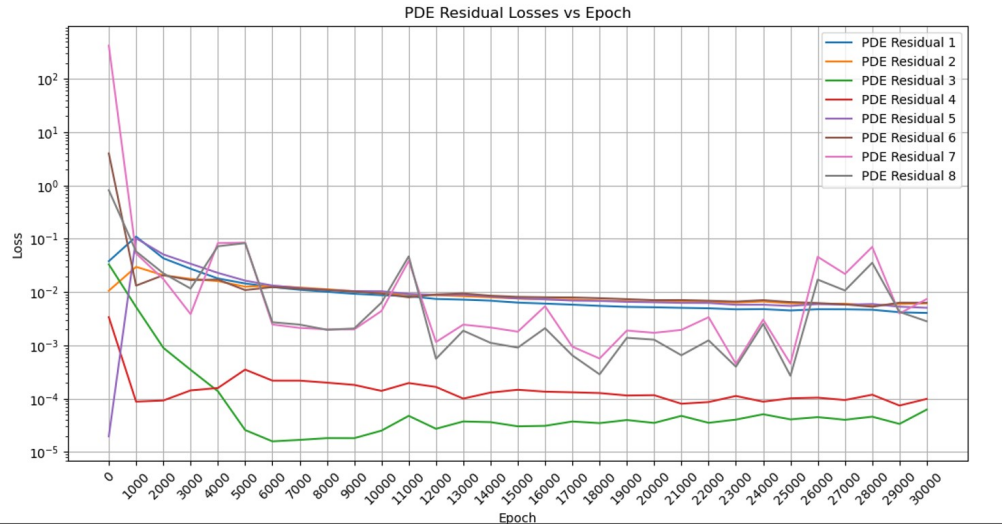
Μια διαδεδομένη μέθοδος για την αντιμετώπιση αυτών των προβλημάτων είναι η επιβολή hard constrains στις αρχικές συνθήκες του προβλήματος. Πρακτικά αυτό σημαίνει να επιβάλουμε έναν μετασχηματισμό της μορφής $N_\varphi(t,\theta)=u_0+\varphi(t)N(t,\theta),\quad t\in\mathbb{R}$ στην έξοδο του δικτύου, όπου u_0 οι αρχικές συνθήκες , $N(t,\theta)$ η αρχική έξοδος του δικτύου , $N_\varphi(t,\theta)$ η έξοδος του δικτύου μετά τον μετασχηματισμό και $\varphi(t)$ μια ομαλή συνάρτηση με $\varphi(t_0)=0$ και $\varphi'(t_0)\neq 0$. Τότε από το Universal Approximation Theorem μπορεί να δειχθεί ότι η $N_\varphi(t,\theta)$ μπορεί να προσεγγίσει την λύση με οποιαδήποτε ακρίβεια δεδομένου ότι υπάρχουν αρκετοί νευρώνες ανά στρώμα. (Babni κ.ά., 2025)

Αρχικά εφαρμόσαμε τον μετασχηματισμό $N_\varphi(t,\theta)=u_0+t\,N(t,\theta)$ σε ένα δίκτυο με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα. Επίσης θέσαμε όλα τα βάρη ίσα με 1 ώστε να παρατηρήσουμε την συμπεριφορά του δικτύου. Χρησιμοποιήσαμε adam optimizer με ρυθμό μάθησης 0.001 και εκπαιδεύσαμε για 30000 εποχές.

Στα παρακάτω σχήματα δίνονται οι συνολικές απώλειες καθώς και οι απώλειες ανά όρο των υπολοίπων της Δ.Ε. Επίσης φαίνονται οι τροχιές των δύο σωμάτων.



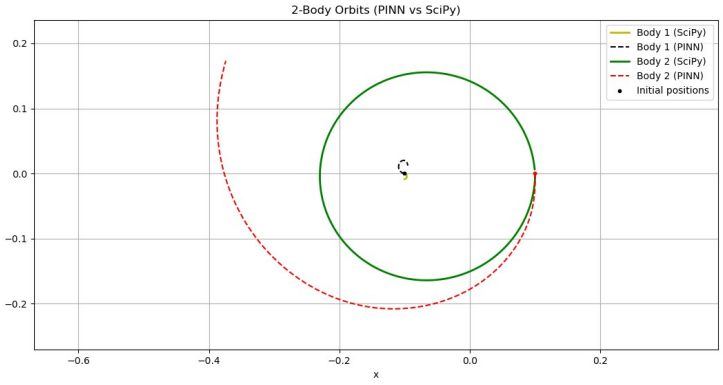
Σχήμα 21: hard-constrains-1st order 2 body pinn: Συνολικές απώλειες



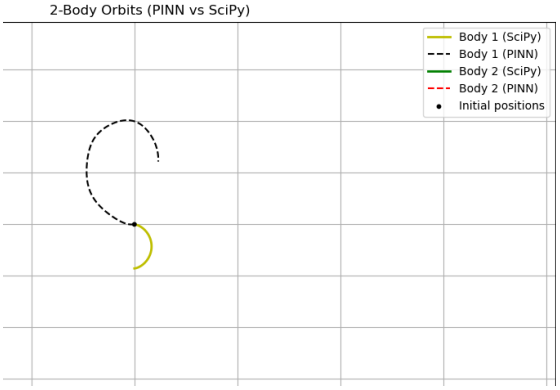
Σχήμα 22: hard-constrains-1st order 2 body pinn: Απώλειες ανά όρο

Από το διάγραμμα απωλειών ανά όρο των υπολοίπων της Δ.Ε (σχήμα 22) παρατηρούμε ότι τα υπόλοιπα 1 και 2 που αφορούν τους όρους των ταχυτήτων του σώματος 1 είναι δύο τάξεις μεγέθους μεγαλύτερα από τους όρους των επιταχύνσεων (υπόλοιπα 3 και 4). Το ίδιο ισχύει εν γένει και για το σώμα 2 όμως η διαφορά είναι μικρότερη. Μια εξήγηση είναι ότι στο πρωτοβάθμιο σύστημα η φυσική περιέχεται στα υπόλοιπα των επιταχύνσεων τα οποία έχουν την μορφή $a_{xi}=\dot{v}_{xi}=G\,m_j\frac{(x_j-x_i)}{(r_{12}+\varepsilon)^3}$. Τα υπόλοιπα των ταχυτήτων έχουν την μορφή $\frac{dx_i}{dt}-v_{xi}$ και απλά δηλώνουν ότι οι ταχύτητες είναι οι παράγωγοι των θέσεων.

Αυτή η διαφορά μεταξύ ταχυτήτων και επιταχύνσεων είναι δέκα φορές εντονότερη στο σώμα 1 που όπως φαίνεται και στα παρακάτω σχήματα έχει τροχιά αντίστροφης κλίσης από την αναμενόμενη.



Σχήμα 24: hard-constrains-1st order 2 body pinn: Τροχιές

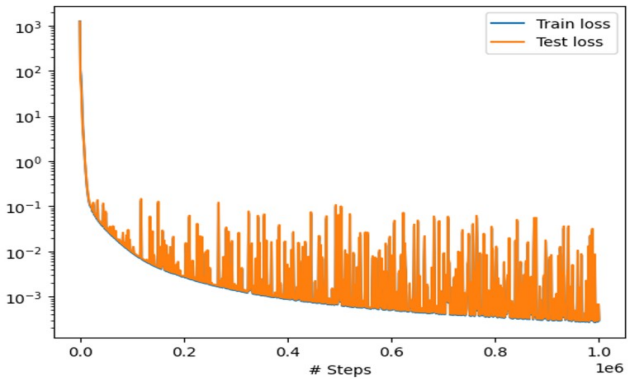


Σχήμα 23: hard-constrains-1st order 2 body pinn: Λεπτομέρεια τροχιάς σώματος 1

Οι παραπάνω παρατηρήσεις μας οδηγούν να ενισχύσουμε σημαντικά τα βάρη στους όρους των ταχυτήτων ώστε τουλάχιστον σε κάθε σώμα ξεχωριστά τα σφάλματα στις ταχύτητες και στις επιταχύνσεις να είναι της ίδιας τάξης. Επίσης από το σχήμα ολικών απωλειών (σχήμα 21) φαίνεται ότι απαιτείται μικρότερος ρυθμός μάθησης για να πετύχουμε καλύτερη σύγκλιση.

Αλλάζοντας τα βάρη με αναλογία 100:1 υπέρ των όρων των ταχυτήτων και για τις ίδιες εποχές εκπαίδευσης παρατηρήσαμε διόρθωση της τροχιάς του σώματος 1 που ενισχύει την υπόθεση ότι το δίκτυο εκπαιδεύονταν κυρίως στις επιταχύνσεις.

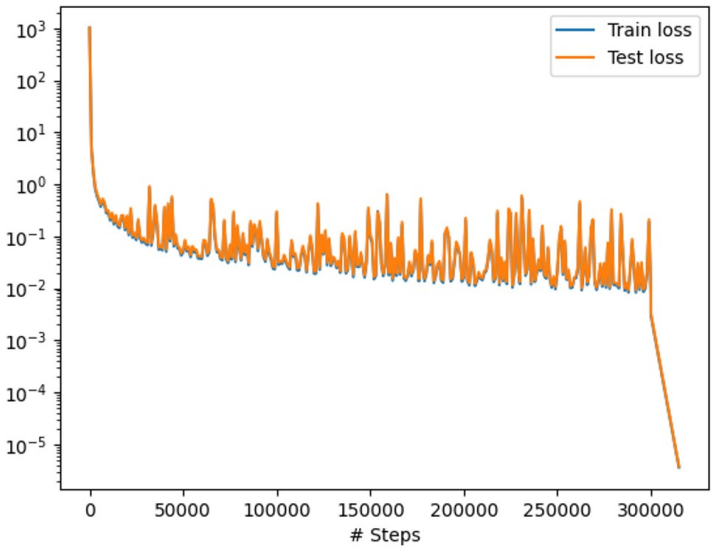
Μια άλλη παρατήρηση είναι ότι η σύγκλιση είναι πολύ αργή και μετά από κάποιο σημείο φαίνεται ότι ο adam optimizer έχει φθάσει στα όριά του. Στο παρακάτω διάγραμμα φαίνονται οι συνολικές απώλειες μετά από 10^6 εποχές. Θέσαμε τα βάρη σε αναλογία 100:1 και ρυθμό μάθησης 10^{-4}.



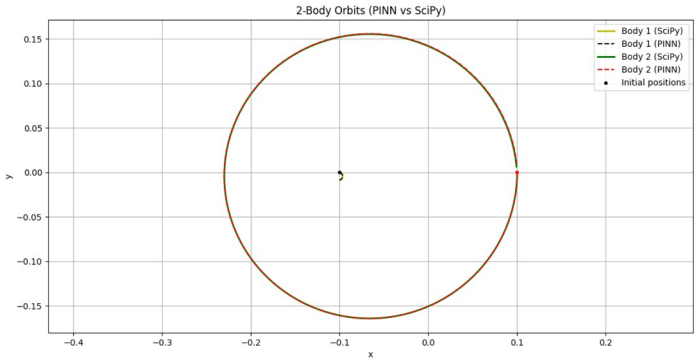
Σχήμα 25: Σύγκλιση 2 body pinn

Στο σχήμα 25 φαίνεται η αργή σύγκλιση του pinn. Επίσης οι πολύ έντονες διακυμάνσεις δείχνουν ότι ο optimizer έχει φθάσει στα όριά του. Μια λύση είναι η χρήση optimizer δεύτερης τάξης μετά από ικανό αριθμό εποχών εκπαίδευσης με τον adam.

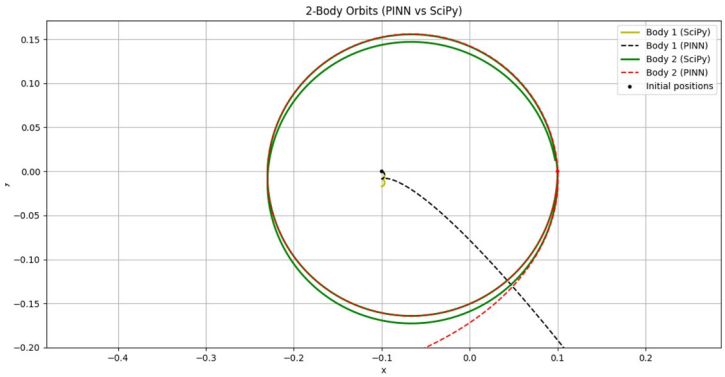
Στα παρακάτω σχήματα φαίνονται τα αποτελέσματα της εκπαίδευσης μετά από 300000 εποχές με adam και κατόπιν εκπαίδευση με τον optimizer L-BFGS. Χρησιμοποιήσαμε δίκτυο με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα. Ο ρυθμός εκπαίδευσης του adam ήταν 10^{-3} και τα βάρη ταχυτήτων-επιταχύνσεων με αναλογία 100:1.



Σχήμα 26: hard-constrains-1st order 2 body pinn-LBFS: Συνολικές απώλειες



Σχήμα 28: hard-constrains-1st order 2 body pinn-L-LBFS: Τροχιές



Σχήμα 27: hard-constrains-1st order 2 body pinn-L-LBFS: Extrapolation

Από το διάγραμμα συνολικών απωλειών (σχ.26) παρατηρούμε ότι ο αλγόριθμος δεύτερης τάξης L-LBFS μας έδωσε σχεδόν τρεις τάξεις μεγέθους καλύτερη ακρίβεια. Επίσης στο σχήμα 28

παρατηρούμε πολύ καλή ταύτιση των τροχιών του pinn (διακεκομένες γραμμές) με την αριθμητική λύση. Στο σχήμα 29 το pinn μας δίνει και μια πρόβλεψη (extrapolation). Αν και η εκπαίδευση ήταν για περίπου μια περίοδο το pinn δίνει σωστή λύση και για χρονικό διάστημα μεγαλύτερο της μιας περιόδου.

Το συμπέρασμα είναι ότι το pinn για το πρόβλημα των δύο σωμάτων με την επιβολή hard constrains και κατάλληλα βάρη – ρυθμό εκπαίδευσης δίνει καλά αποτελέσματα. Επίσης η χρήση του αλγόριθμου δεύτερης τάξης βελτιώνει σε μεγάλο βαθμό τα αποτελέσματα.

Τέλος θα πρέπει να παρατηρήσουμε ότι μια και το σύστημα είναι περιοδικό με γνωστή περίοδο θα μπορούσαμε να επιβάλουμε την περιοδικότητα είτε στον μετασχηματισμό εξόδου (hard constrains) είτε με κάποιο μετασχηματισμό στην είσοδο. Δεν προχωρήσαμε στην περαιτέρω μελέτη του συστήματος διότι ο στόχος είναι το πρόβλημα των τριών σωμάτων που δεν παρουσιάζει περιοδικότητα εν γένει.

Στην επόμενη παράγραφο εξετάζουμε την συμπεριφορά του δευτεροβάθμιου συστήματος που έχει σημαντικά πλεονεκτήματα άλλα μεγάλο κόστος σε μνήμη.

4.1.5 Σύστημα Δ.Ε δεύτερης τάξης με επιβολή των αρχικών συνθηκών (hard constrains)

Για το σύστημα δεύτερης τάξης χρησιμοποιήσαμε τις ίδιες αρχικές θέσεις – ταχύτητες. Επίσης επιβάλαμε τις αρχικές συνθήκες με μετασχηματισμό εξόδου (hard -constrains). Το σημαντικό πλεονέκτημα του συστήματος δεύτερης τάξης είναι ότι έχουμε μόνο τα τέσσερα υπόλοιπα των επιταχύνσεων $a_{xi}=\ddot{x}_i=G\,m_j\frac{(x_j-x_i)}{(r_{12}+\varepsilon)^3}$. Αυτό αποτελεί μεγάλη διαφορά σε σχέση με τα 16 υπόλοιπα του vanilla-pinn και τα οκτώ υπόλοιπα του pinn πρώτης τάξης με hard-contrains.

Το σημαντικό μειονέκτημα του συστήματος δεύτερης τάξης είναι η πολλαπλάσια μνήμη και υπολογιστική ισχύς που απαιτείται διότι ο υπολογισμός δευτέρων παραγώγων με την αυτόματη παραγώγιση (AD) απαιτεί την αποθήκευση στην μνήμη αρχικά των αποτελεσμάτων της πρώτης παραγώγισης. Αυτό μπορεί να οδηγήσει σε εκθετική αύξηση της απαιτούμενης μνήμης.

Παρ όλα αυτά επειδή το δίκτυο έχει τώρα τέσσερις εξόδους (τις θέσεις των δύο σωμάτων στο επίπεδο) και η συνάρτηση σφάλματος μόνο τέσσερα υπόλοιπα η αυξημένη μνήμη και υπολογισμοί τώρα εκτελούνται για λιγότερα υπόλοιπα/εξόδους και αυτό αντισταθμίζει σε κάποιο βαθμό το κόστος σε μνήμη και χρόνο εκτέλεσης.

Οι έξοδοι του δικτύου είναι $y=[x_1,y_1,x_2,y_2]$ που αντιστοιχούν στις θέσεις των σωμάτων 1 και 2 με σχετικές μάζες $m_1=0.99,\ m_2=0.01$.

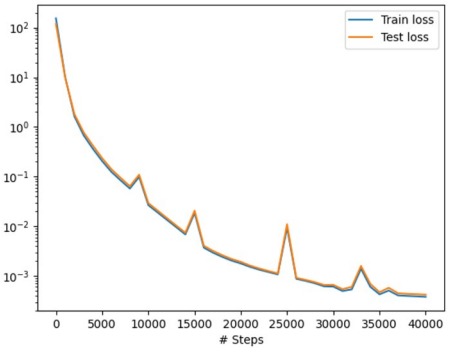
Οι αρχικές συνθήκες για τις θέσεις είναι: $y_0=[x_1(0)=-0.1, y_1(0)=0, x_2(0)=0.1, y_2(0)=0]$ και για τις ταχύτητες $u_0=[\dot{x}_1(0)=0, \dot{y}_1(0)=0, \dot{x}_2(0)=0, \dot{y}_2(0)=-2]$.

Για την επιβολή των αρχικών συνθηκών με hard constrains χρησιμοποιούμε τον μετασχηματισμό εξόδου $N_\varphi(t,\theta)=y_0+t\,u_0+t^2N(t,\theta)$ έτσι ώστε $N_\varphi(0,\theta)=y_0$ και $\frac{d\,N_\varphi(0,\theta)}{dt}=u_0$.

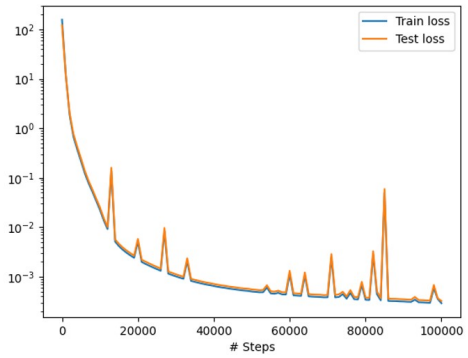
Επίσης χρησιμοποιήσαμε adamW optimizer διότι στην συγκεκριμένη περίπτωση δίνει καλύτερα αποτελέσματα από τον adam. Η παρούσα έκδοση της deepxde δεν έχει τον adamW σαν αυτόματη επιλογή και χρειάστηκε να γράψουμε τον κώδικα που καλεί τον adamW από την βιβλιοθήκη tensorflow. Ο κώδικας βρίσκεται στο παράρτημα Γ.

Το δίκτυο που χρησιμοποιήσαμε έχει 3 κρυφά στρώματα με 64 νευρώνες ανά στρώμα όπως και στα προηγούμενα παραδείγματα. Ο αλγόριθμος adamW είχε παραμέτρους learning rate 10^{-4} και

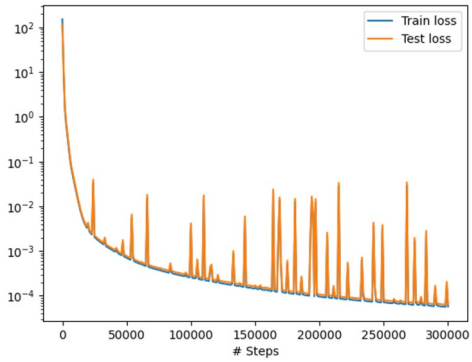
weight decay 10^{-5}. Το pinn εκπαιδεύτηκε για χρόνο t=0.4 που είναι λίγο μικρότερος από μία περίοδο. Παρακάτω δίνονται τα αποτελέσματα για 40000, 100000 και 300000 εποχές εκπαίδευσης.



Σχήμα 29: 2nd order pinn for 2 body - 40000 εποχές

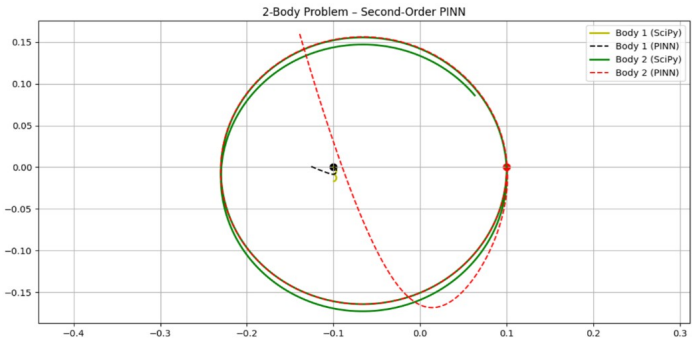


Σχήμα 30: 2nd order pinn for 2 body - 100000 εποχές

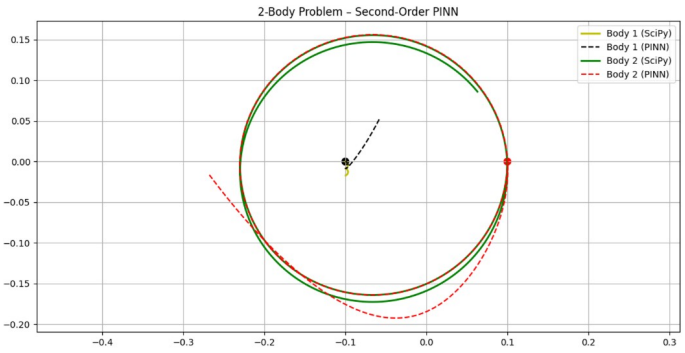


Σχήμα 31: 2nd order pinn for 2 body - 300000 εποχές

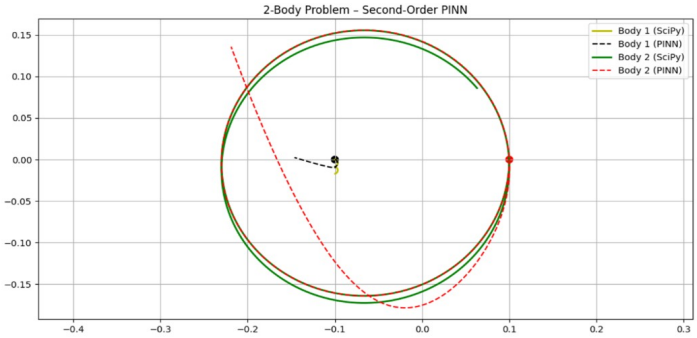
Από τα παραπάνω διαγράμματα ολικών απωλειών φαίνεται ότι έχουμε μια αργή αλλά σταθερή σύγκλιση. Ειδικά το τελευταίο διάγραμμα (σχ.31) δείχνει ότι το pinn μπορεί να εκπαιδευτεί περαιτέρω.



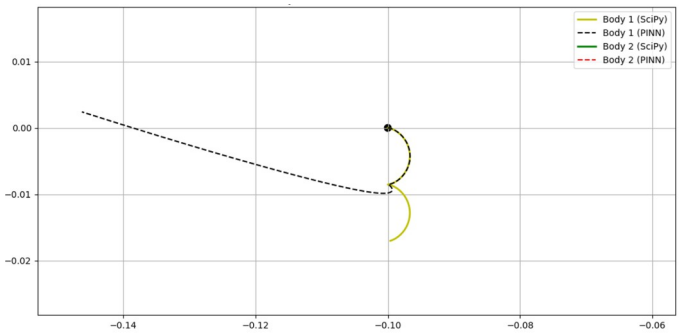
Σχήμα 32: 2nd order pinn for 2 body -extrapolation - 40000 εποχές



Σχήμα 33: 2nd order pinn for 2 body -extrapolation - 100000 εποχές



Σχήμα 34: 2nd order pinn for 2 body -extrapolation - 300000 εποχές



Σχήμα 35: 2nd order pinn for 2 body -extrapolation - 300000 εποχές - μεγέθυνση τροχιάς

Από τα σχήματα 32-34 φαίνεται πως το pinn δίνει μια σωστή πρόβλεψη για ένα μικρό κλάσμα της περιόδου. Αν παρατηρήσουμε μόνο το σώμα μικρής μάζας (μεγάλη τροχιά) δεν φαίνεται μεγάλη διαφορά ενώ οι περίοδοι εκπαίδευσης τριπλασιάζονται σχεδόν από το ένα σχήμα στο επόμενο. Όμως παρατηρώντας το σώμα μεγάλης μάζας φαίνονται σημαντικές διαφορές στις τροχιές. Ειδικά στο σχήμα 35 που δείχνει λεπτομέρεια της τροχιάς του σώματος μεγάλης μάζας φαίνεται ότι μετά από 300000 εποχές εκπαίδευσης το pinn αρχίζει να προβλέπει σωστά την τροχιά του. Η τροχιά του σώματος μεγάλης μάζας έχει πολύ πιο απότομη κλίση όταν περνάμε από την πρώτη περίοδο στην δεύτερη λόγω του σχήματος “ωμέγα” που φαίνεται στο σχήμα.

4.1.6 Συμπεράσματα από τα πειράματα με το σύστημα των δύο σωμάτων

Αν και είναι σύνηθες κατά την αριθμητική επίλυση ενός συστήματος Δ.Ε να μετατρέπουμε ένα σύστημα δεύτερης τάξης σε σύστημα πρώτης τάξης αυτή η πρακτική στα pinns δεν δίνει πάντα βέλτιστα αποτελέσματα. Στα pinns κυρίαρχο ρόλο έχουν οι όροι απωλειών που όσο αυξάνεται το πλήθος τους τόσο πιο δύσκολη είναι η σύγκλιση. Επίσης οι όροι απωλειών θα πρέπει να είναι κατά το δυνατόν ίδιας τάξης ώστε η μάθηση να είναι ομαλή. Αυτό το πρόβλημα σε κάποιες περιπτώσεις μπορεί να αντιμετωπισθεί με κατάλληλα βάρη στην συνάρτηση απωλειών όμως όπως δείξαμε υπάρχουν περιπτώσεις όπου αυτό είναι πρακτικά πολύ δύσκολο αν όχι αδύνατο. Αυτό οδηγεί σε αλγόριθμους μάθησης δεύτερης τάξης όπως ο L-BFGS που απαιτούν πολλαπλάσια μνήμη και χρόνο για την εκπαίδευση.

Μια άλλη σημαντική δυσκολία των pinns που τα διαφοροποιεί από τις μεθόδους αριθμητικής επίλυσης είναι η δυσκολία τους να είναι συνεπή με τις αρχικές συνθήκες ειδικά σε προβλήματα με μεγάλο πλήθος αρχικών συνθηκών.

Από τα πειράματα με το σύστημα δύο σωμάτων φαίνεται ότι η επιβολή των αρχικών συνθηκών με έναν μετασχηματισμό στην έξοδο του δικτύου (hard constrains) βελτιώνει σημαντικά την σύγκλιση.

Τέλος παρατηρήσαμε ότι αν και πιο απαιτητικό σε μνήμη το σύστημα δεύτερης τάξης δίνει πολύ καλύτερα αποτελέσματα. Το παράδοξο ίσως είναι ότι αν και το σύστημα δεύτερης τάξης θεωρείται απαιτητικό σε μνήμη και σε χρόνο εν τέλει είναι εκτός από πιο ακριβές γρηγορότερο σε μάθηση διότι δεν χρειάζεται να χρησιμοποιήσουμε αλγόριθμους μάθησης δεύτερης τάξης που είναι ιδιαίτερα χρονοβόροι και δύσκολο να βελτιστοποιηθούν.

4.2 Το πρόβλημα των τριών σωμάτων

4.2.1 Περιγραφή του προβλήματος

Το πρόβλημα των τριών σωμάτων είναι ένα διάσημο μη-επιλύσιμο αναλυτικά πρόβλημα από την εποχή του Newton. Το πρόβλημα αρχικά μελετήθηκε από τον Newton στο έργο του *Principia (Newton, 1687)* και αφορούσε την κίνηση του συστήματος Γης-Σελήνης γύρω από τον Ήλιο.

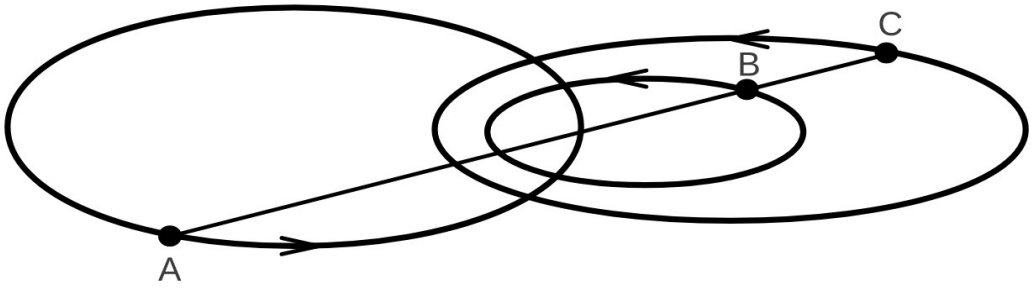
Ο *Poincaré* στο τρίτομο έργο του *Les Méthodes nouvelles de la mécanique céleste* (Poincaré, 1892) έδειξε ότι οι τροχιές είναι απρόβλεπτες και ανακάλυψε το φαινόμενο που σήμερα ονομάζουμε χάος.

Επειδή το πρόβλημα είναι χαοτικό, μικρές διαφορές στις αρχικές συνθήκες προκαλούν σημαντικές μεταβολές στις τροχιές των σωμάτων. Ιδιαίτερο ενδιαφέρον έχουν οι τροχιές που είναι περιοδικές. Οι περιοδικές τροχιές μπορούν να κατηγοριοποιηθούν σε οικογένειες τροχιών με βάση την τοπολογική μέθοδο του Montgomery. (Montgomery, 1998)

Είναι εντυπωσιακό ότι μόνο τρεις οικογένειες περιοδικών λύσεων είχαν βρεθεί 300 χρόνια μετά την αρχική αναφορά στο έργο του Newton. Η πρώτη είναι η οικογένεια Euler-Lagrange η δεύτερη είναι η ΒΗΗ που βρέθηκε από τους Broucke (1975), Hadjidemetriou (1975) and Hénon (1976) και η τρίτη είναι η οικογένεια τροχιών σχήματος 8 (figure-eight family) που βρέθηκε με αριθμητικές μεθόδους από τον Moore (1993). Το 2013 οι Suvakov και Dmitrasinovic βρήκαν 11 νέες οικογένειες για σώματα ίσης μάζας με χρήση προσομοίωσης σε υπολογιστή και μέχρι σήμερα έχουν βρεθεί χιλιάδες νέες οικογένειες τροχιών με χρήση υπολογιστών.

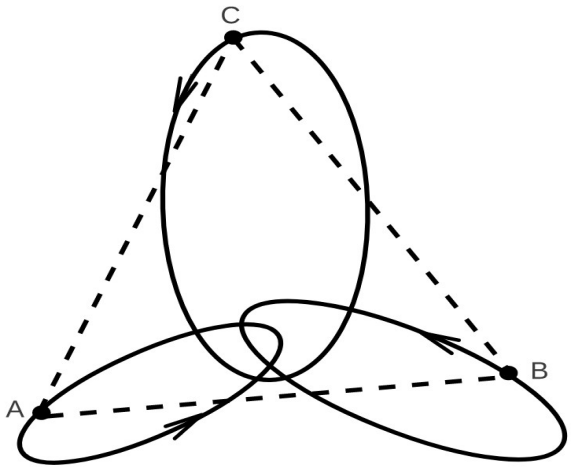
Η κυριότερη δυσκολία στην εύρεση περιοδικών τροχιών με μεθόδους προσομοίωσης σε υπολογιστή είναι η ευαισθησία του συστήματος στις αρχικές συνθήκες (butterfly-effect) που κάνει την προσομοίωση για μεγάλο χρονικό διάστημα μη αξιόπιστη λόγω σφαλμάτων. Για την αντιμετώπιση αυτού του προβλήματος ο Liao πρότεινε μια νέα μέθοδο αριθμητικής προσομοίωσης για χαοτικά συστήματα που ονομάζεται clean numerical simulation (CNS). (LIAO κ.ά., 2021)

Οι πρώτες περιοδικές τροχιές βρέθηκαν από τους Euler (1767) και Lagrange (1772) . Ο Euler θεώρησε τρία σώματα που αρχικά βρίσκονται στην ίδια ευθεία και έδειξε ότι για κατάλληλες συνθήκες τα τρία σώματα θα παρέμεναν στην ίδια ευθεία εκτελώντας το καθένα μία ελλειπτική τροχιά γύρω από το κέντρο μάζας του συστήματος.



Σχήμα 36: Περιοδικές τροχιές του Euler (Musielak & Quarles, 2014)

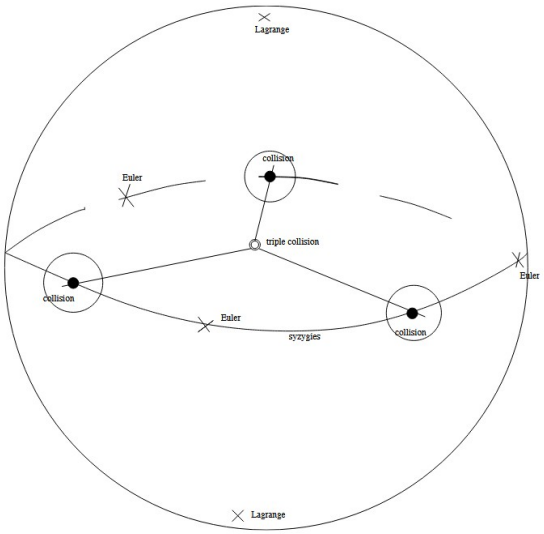
Στο σχήμα 36 φαίνεται η περιοδική λύση του Euler. Αν ο λόγος AB/BC έχει συγκεκριμένη τιμή που εξαρτάται από τις μάζες τότε για κατάλληλες αρχικές συνθήκες τα τρία σώματα εκτελούν περιοδική κίνηση ενώ παραμένουν διαρκώς στην ίδια ευθεία. Θα πρέπει να σημειωθεί ότι η λύση του Euler είναι ασταθής σε μικρές μετατοπίσεις, ένα μικρό σφάλμα στην αρχική διαμόρφωση αυξάνεται με τον χρόνο καταστρέφοντας την περιοδικότητα. (Musielak & Quarles, 2014)



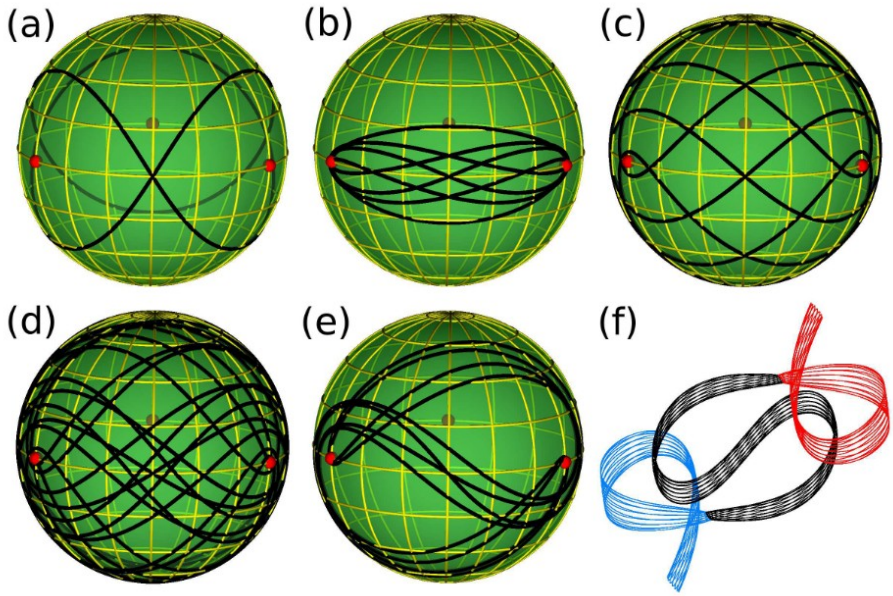
Σχήμα 37: Περιοδική λύση του Lagrange
(Musielak & Quarles, 2014)

Στο σχήμα 37 φαίνεται η περιοδική λύση του Lagrange. Η αρχική διαμόρφωση είναι ένα ισόπλευρο τρίγωνο με τα σώματα στις κορυφές του. Ο Lagrange έδειξε ότι για κατάλληλες αρχικές συνθήκες η αρχική διαμόρφωση διατηρείται και τα σώματα εκτελούν ελλειπτικές περιοδικές τροχιές. Καθώς κινούνται τα σώματα το τρίγωνο που σχηματίζουν αλλάζει μέγεθος και προσανατολισμό αλλά παραμένει ισόπλευρο. (Musielak & Quarles, 2014)

Οι τροχιές των τριών σωμάτων μπορούν να παρασταθούν σε μία σφαίρα γνωστή ως shape-sphere που προτάθηκε από τον Montgomery. Η ιδέα είναι ότι τα τρία σώματα σχηματίζουν σε κάθε στιγμή ένα τρίγωνο. Χρησιμοποιώντας τις φυσικές συμμετρίες του προβλήματος μπορούμε να αντιληφθούμε ότι ο ευκλείδειος χώρος δεν είναι ο καταλληλότερος για την περιγραφή του προβλήματος. Για παράδειγμα αν μετατοπίσουμε ή στρέψουμε το τρίγωνο των τροχιών η φυσική δεν αλλάζει αν και τα τρίγωνα περιγράφονται από διαφορετικές συντεταγμένες στον ευκλείδειο χώρο. Έτσι ο ευκλείδειος χώρος δεν βοηθά ώστε να γενικέψουμε τις λύσεις με βάση τις συμμετρίες του προβλήματος. Στον χώρο της shape-sphere κάθε σημείο αντιστοιχεί σε μια κλάση τριγώνων. Για παράδειγμα τα άπειρα τρίγωνα που προκύπτουν από μετατοπίσεις ,στροφές και αλλαγή κλίμακας τυχαίου τριγώνου αντιστοιχούν σε ένα σημείο. Οι πόλοι της σφαίρας αντιστοιχούν σε ισόπλευρα τρίγωνα (τροχιές Lagrange) και ο ισημερινός σε διαμορφώσεις όπου όλα τα σημεία είναι στην ίδια ευθεία (τροχιές Euler). (Montgomery, 2014)



Σχήμα 38: Shape - sphere
(Montgomery, 2014)



Σχήμα 39: Περιοδικές τροχιές στην Shape-Sphere (Šuvakov & Dmitrašinović, 2013)

Στο σχήμα 39 φαίνονται διάφορες περιοδικές τροχιές στον χώρο της shape-sphere:

(a) Τροχιές σχήματος οκτώ (figure-eight)

(b) Τροχιές πεταλούδας (Class I.A butterfly I orbit)

(c) Class I.B moth I orbit

(d) Class II.B yarn orbit

(e) Τροχιά Γιν και Γιανγκ I (Class II.C yin-yang I orbit)

(f) Η τροχιά Γιν και Γιανγκ II στον ευκλείδειο χώρο. (Šuvakov & Dmitrašinović, 2013)

4.2.2 Μαθηματική διατύπωση και διαστατική ανάλυση

Το πρόβλημα αφορά τρία σημειακά σώματα μάζας m_i που αλληλεπιδρούν μόνο με την βαρυτική έλξη. Έστω $\vec{r}_i \quad i=1,2,3$ τα διανύσματα θέσης των σωμάτων σε ένα σύστημα καρτεσιανών συντεταγμένων. Τότε οι εξισώσεις κίνησης δίνονται από τον 2ο νόμο του Newton:

$$\ddot{\vec{r}}_i=-G\sum_{j=1;\,j\neq i}^{3} m_j\frac{(\vec{r}_i-\vec{r}_j)}{|\vec{r}_i-\vec{r}_j|^3} \quad i,j=1,2,3 \quad και \quad \vec{r}_i=(x_i,y_i,z_i)$$

Για να απλοποιήσουμε το πρόβλημα μπορούμε να θεωρήσουμε κινήσεις περιορισμένες σε ένα επίπεδο (x-y). Αυτό είναι πάντα δυνατόν αρκεί οι αρχικές ταχύτητες των σωμάτων να βρίσκονται στο ίδιο επίπεδο. Τότε από την διατήρηση της στροφορμής του συστήματος τα σώματα θα παραμένουν συνέχεια στο ίδιο αρχικό επίπεδο.

Η ολική στροφορμή του συστήματος δίνεται από την σχέση $\vec{L}=\sum_i m_i\vec{r}_i\times\vec{u}_i$ και είναι διατηρούμενη ποσότητα $\frac{d\vec{L}}{dt}=0$. Επομένως αν αρχικά οι ταχύτητες είναι στο ίδιο επίπεδο η ολική στροφορμή θα είναι κάθετη σε αυτό το επίπεδο και αφού διατηρείται οι ταχύτητες και οι τροχιές θα βρίσκονται διαρκώς στο αρχικό επίπεδο.

Με την απλοποίηση σε δύο διαστάσεις θα έχουμε 6 εξισώσεις 2ης τάξης:

$$\ddot{x}_i=-G\sum_{j=1;\,j\neq i}^{3} m_j\frac{(x_i-x_j)}{r_{ij}^{\,3}},\quad \ddot{y}_i=-G\sum_{j=1;\,j\neq i}^{3} m_j\frac{(y_i-y_j)}{r_{ij}^{\,3}},\quad r_{ij}=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\quad i,j=1,2,3.$$

(Hysa κ.ά., 2025)

Το πρόβλημα μπορεί να γραφεί και με 12 εξισώσεις 1ης τάξης:

$$\dot{x}_i=v_{xi},\quad \dot{v}_{xi}=-G\sum_{j=1;\,j\neq i}^{3} m_j\frac{(x_i-x_j)}{r_{ij}^{\,3}},\quad \dot{y}_i=v_{yi},\quad \dot{v}_{yi}=-G\sum_{j=1;\,j\neq i}^{3} m_j\frac{(y_i-y_j)}{r_{ij}^{\,3}}\quad i,j=1,2,3.$$

Ο Lagrange χρησιμοποιώντας τους νόμους διατήρησης ορμής, στροφορμής και ενέργειας απλοποίησε το σύστημα. Χρησιμοποιώντας τους νόμους διατήρησης για την επίπεδη κίνηση μπορούμε να μειώσουμε τις εξισώσεις 1ης τάξης επιλέγοντας το κέντρο μάζας ακίνητο διαρκώς (λόγω διατήρησης ορμής) και επίσης από την διατήρηση της στροφορμής (ελευθερία στροφών του συστήματος συντεταγμένων πάνω στο επίπεδο) μπορούμε να εργαστούμε σε ένα σταθερά στρεφόμενα σύστημα. (Krishnaswami & Senapati, 2019)

Πάντως για την αριθμητική επίλυση δεν είναι απαραίτητο να κάνουμε αυτούς τους μετασχηματισμούς. Μπορούμε για καλύτερη διαφάνεια των εξισώσεων να δουλέψουμε στο αρχικό πρόβλημα.

Σχετικά με την διαστατική ανάλυση το πρόβλημα των τριών σωμάτων δεν έχει διαφορά από το σύστημα δύο σωμάτων που αναλύσαμε στην παράγραφο 4.1.2. Έτσι θέτοντας την σταθερά παγκόσμιας έλξης $G=1$ έχουμε ένα αδιάστατο πρόβλημα αρκεί να επιλέξουμε κατάλληλες κλίμακες για τις μάζες και τις αποστάσεις. Συνήθως στο πρόβλημα των τριών σωμάτων θέτουμε σαν κλίμακα μάζας M την μάζα του σώματος μεγαλύτερης μάζας. Έτσι αν έχουμε σώματα ίσης μάζας οι αδιάστατες μάζες τους θα είναι $\mu_1=\frac{m_1}{M}=\mu_2=\frac{m_2}{M}=\mu_3=\frac{m_3}{M}=1$. Για τις αποστάσεις μπορούμε να επιλέξουμε είτε την μεγαλύτερη αρχική απόσταση των σωμάτων είτε κάποια άλλη κατάλληλη απόσταση ανάλογα με το πρόβλημα που επιλύουμε. Εδώ θα πρέπει να αναφέρουμε ότι στο πρόβλημα των τριών σωμάτων δεν έχουμε ούτε κλειστές ούτε περιοδικές τροχιές εν γένει. Επομένως τα σώματα μπορεί να πλησιάζουν ή να απομακρύνονται απεριόριστα.

4.2.3 Περιοδικές τροχιές Euler – Σύστημα 1ης τάξης

Χρησιμοποιήσαμε το σύστημα με τις 12 εξισώσεις 1ης τάξης και επιβάλαμε τις αρχικές συνθήκες (hard-constrains) ώστε να μην έχουμε πολλούς όρους σφαλμάτων.

Το pinn είχε 12 εξόδους για τις θέσεις και τις ταχύτητες των τριών σωμάτων

$$y=[x_1, y_1, v_{x1}, v_{y1}, x_2, y_2, v_{x2}, v_{y2}, x_3, y_3, v_{x3}, v_{y3},]$$

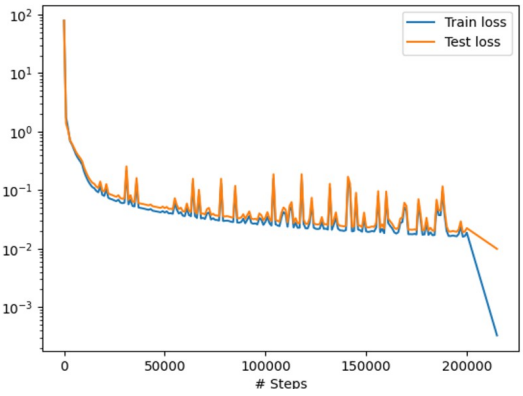
Οι μάζες των σωμάτων ήταν ίσες $m_1=m_2=m_3=1.$.

Οι αρχικές συνθήκες για τα τρία σώματα ήταν

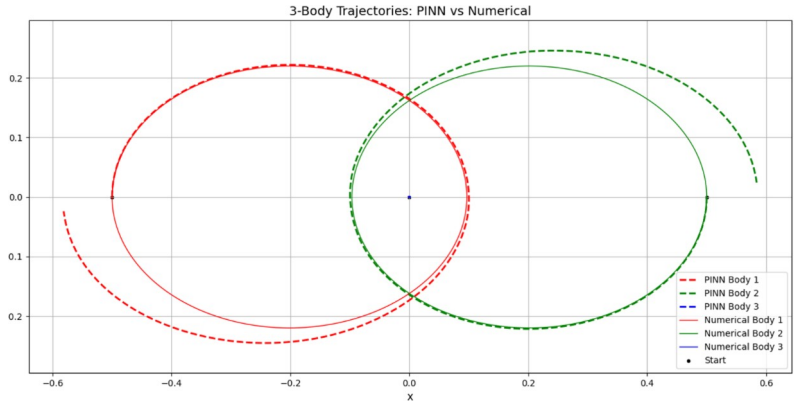
$$y_0=[x_1(0)=-0.5, y_1(0)=0.0, v_{x1}(0)=0.0, v_{y1}(0)=0.9,$$
$$x_2(0)=0.5, y_2(0)=0.0, v_{x2}(0)=0.0, v_{y2}(0)=-0.9,$$
$$x_3(0)=0.0, y_3(0)=0.0, v_{x3}(0)=0.0, v_{y3}(0)=0.0]$$

Παρατηρούμε ότι το σώμα 3 είναι ακίνητο ανάμεσα στα σώματα 1 και 2. Η μη αναμενόμενη αρίθμηση έγινε ώστε να είναι εύκολο να συγκρίνουμε τα υπόλοιπα των σωμάτων 1 και 2 που περιστρέφονται γύρω από το σώμα 1. Ο μετασχηματισμός εξόδου για την επιβολή των αρχικών συνθηκών ήταν $y_\varphi(t,\theta)=y_0+t\,y(t,\theta)$.

Χρησιμοποιήθηκε ένα δίκτυο με 3 κρυφά στρώματα και 128 νευρώνες ανά στρώμα και βάρη ίσα με 1 για όλα τα υπόλοιπα. Αρχικά το δίκτυο εκπαιδεύτηκε με adam optimizer με learning rate 10^{-4} για 200000 εποχές και κατόπιν χρησιμοποιήθηκε ο αλγόριθμος 2ης τάξης L-BFGS για επιπλέον 15000 εποχές.



Σχήμα 40: Τροχιές Euler -Σύστημα 1ης τάξης - Συνολικές απώλειες

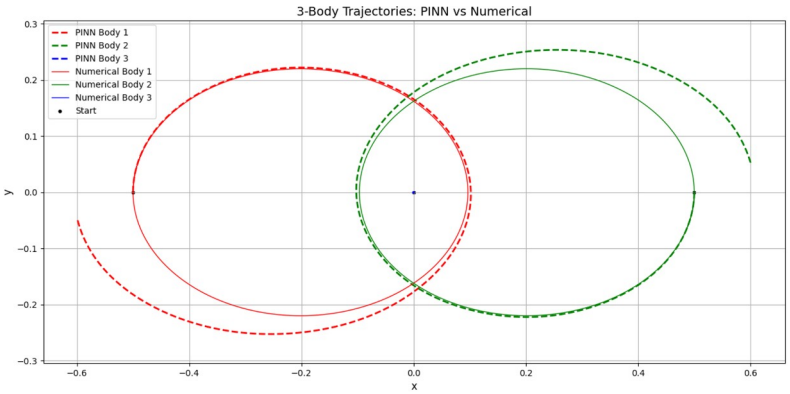


Σχήμα 41: Τροχιές Euler -Σύστημα 1ης τάξης - Τροχιές

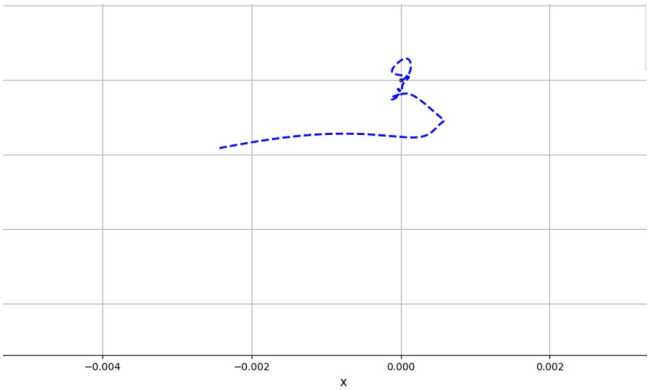
Από το σχήμα 40 παρατηρούμε ότι η εκπαίδευση με τον αλγόριθμο adam είναι πολύ αργή ενώ ο αλγόριθμος L-BFGS αυξάνει την ακρίβεια κατά δύο τάξεις. Μία άλλη παρατήρηση είναι ότι το test-loss δεν ακολουθεί το train-loss κατά την διάρκεια εκπαίδευσης με τον L-BFGS . Αυτό πιθανόν συμβαίνει διότι έχουμε 64 σημεία για την εκπαίδευση του μοντέλου και 100 για τα test losses. Αν αυξήσουμε τα σημεία εκπαίδευσης (collocation points) πριν την χρήση του L-BFGS δεν παρατηρούμε πλέον το φαινόμενο.

Στο σχήμα 41 παρατηρούμε ότι οι τροχιές έχουν πολύ καλή ταύτιση κατά την πρώτη ημιπερίοδο και κατόπιν το σφάλμα αυξάνεται σταθερά. Από τα πειράματα προέκυψε ότι η εικόνα του σχήματος 41 είναι δύσκολο να καλυτερέψει αλλάζοντας τον ρυθμό μάθησης και τα βάρη. Αν και η διαμόρφωση που χρησιμοποιήσαμε είναι η πιο απλή περίπτωση περιοδικών τροχιών Euler φαίνεται ότι το pinn με εξισώσεις 1ης τάξης έχει δυσκολία στην σύγκλιση. Πιθανόν αυτό να οφείλεται στο ότι οι τροχιές Euler είναι ασταθείς. Μια μικρή απόκλιση στην τροχιά του διαρκώς ακίνητου σώματος (σώμα 3 στο σημείο 0,0)μπορεί να προκαλέσει σημαντικό πρόβλημα στην σύγκλιση του pinn.

Στα σχήματα 42 και 43 παρατηρούμε τα αποτελέσματα ενός παρόμοιου πειράματος με 600000 εποχές εκπαίδευση και χρήση του αλγόριθμου L-BFGS. Στο σχήμα 43 παρατηρούμε σε μεγέθυνση την τροχιά του “ακίνητου” σώματος που μετά από κάποιο χρονικό διάστημα αρχίζει να απομακρύνεται από την αρχική θέση (0,0).



Σχήμα 42: Τροχιές Euler -Σύστημα 1ης τάξης: Τροχιές που αποκλίνουν



Σχήμα 43: Τροχιές Euler -Σύστημα 1ης τάξης: Λεπτομέρεια τροχιάς "ακίνητου" σώματος

4.2.4 Περιοδικές τροχιές Euler – Σύστημα 2ης τάξης

Για να βελτιώσουμε τα αποτελέσματα που παρατηρήσαμε στο σύστημα 1ης τάξης χρησιμοποιήσαμε το σύστημα 2ης τάξης Δ.Ε που όπως έχουμε αναφέρει έχει το πλεονέκτημα των λιγότερων υπόλοιπων στην συνάρτηση σφάλματος με το κόστος της αυξημένης μνήμης και χρόνου εκτέλεσης. Χρησιμοποιήσαμε τις ίδιες αρχικές συνθήκες και μάζες των σωμάτων με το σύστημα 1ης τάξης ώστε να έχουμε εύκολη σύγκριση των αποτελεσμάτων.

Το pinn είχε 6 εξόδους για τις θέσεις των τριών σωμάτων $y=[x_1, y_1, x_2, y_2, x_3, y_3]$.

Οι μάζες των σωμάτων ήταν ίσες $m_1=m_2=m_3=1.$.

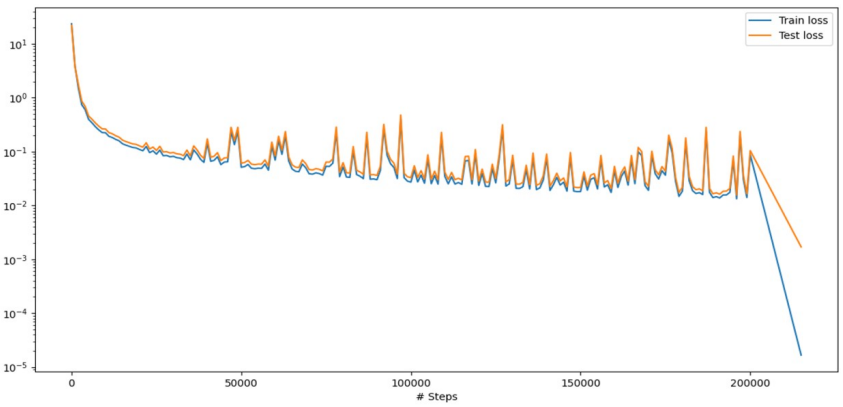
Οι αρχικές συνθήκες για τις θέσεις ήταν:

$$y_0=[x_1(0)=-0.5, y_1(0)=0, x_2(0)=0.5, y_2(0)=0, x_3(0)=0, y_3(0)=0]$$

και για τις ταχύτητες $u_0=[\dot{x}_1(0)=0, \dot{y}_1(0)=0.9, \dot{x}_2(0)=0, \dot{y}_2(0)=-0.9, \dot{x}_3(0)=0, \dot{y}_3(0)=0]$.

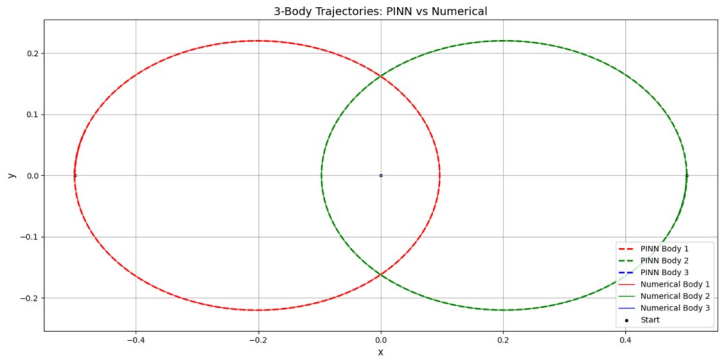
Επιβάλαμε τις αρχικές συνθήκες με τον μετασχηματισμό εξόδου $y_\varphi(t,\theta)=y_0+t\,u_0+t^2\,y(t,\theta)$

Αρχικά εκπαιδεύσαμε ένα δίκτυο με 3 κρυφά στρώματα και 128 νευρώνες ανά στρώμα με αλγόριθμο adam , ρυθμό μάθησης 10^{-4} και βάρη ίσα με 1 για 200000 εποχές. Και κατόπιν χρησιμοποιήσαμε τον αλγόριθμο L-BFGS ώστε να έχουμε ακριβώς την ίδια διαμόρφωση - υπερπαραμέτρους με την περίπτωση του συστήματος πρώτης τάξης.

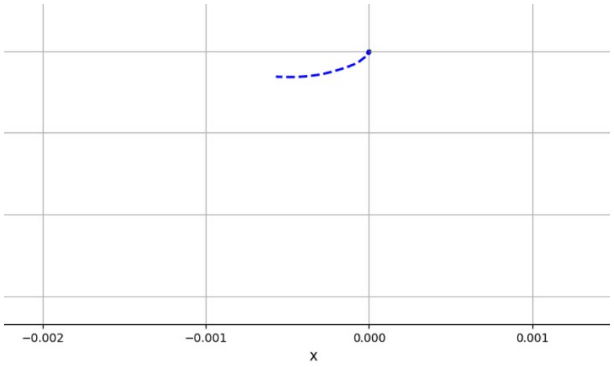


Σχήμα 44: Τροχιές Euler -Σύστημα 2ης τάξης: Απώλειες δίκτυο 3x128

Παρατηρούμε ότι ο αλγόριθμος δεύτερης τάξης δίνει επιπλέον ακρίβεια για τρεις τάξεις μεγέθους.



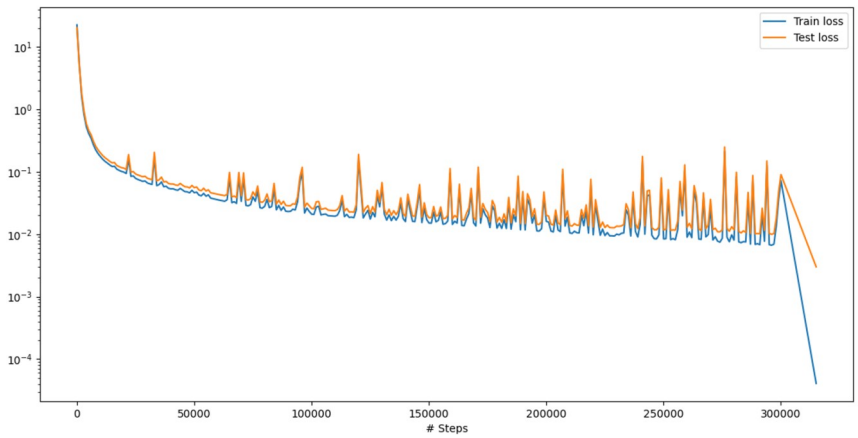
Σχήμα 45: Τροχιές Euler -Σύστημα 2ης τάξης: Τροχίες δίκτυο 3x128



Σχήμα 46: Τροχιές Euler -Σύστημα 2ης τάξης: Λεπτομέρια τροχιάς "ακίνητου" σώματος

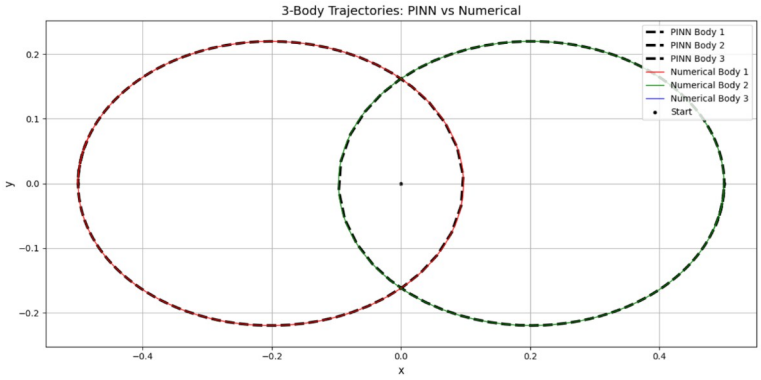
Στο σχήμα 45 φαίνεται η πολύ καλή ταύτιση των τροχιών με την αριθμητική λύση. Επίσης στο σχήμα 46 παρατηρούμε ότι η τροχιά του “ακίνητου” σώματος έχει αρκετά μικρότερη απόκλιση σε σχέση με το αντίστοιχο διάγραμμα του συστήματος 1ης τάξης.

Επειδή ο αλγόριθμος L-BFGS είναι χρονοβόρος σε δίκτυα με πολλούς νευρώνες χρησιμοποιήσαμε ένα δίκτυο με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα. Επίσης χρησιμοποιήσαμε των αλγόριθμό adamW διότι παρατηρήσαμε ότι έχει καλύτερη σύγκλιση στην περίπτωση των pinns με συστήματα 2ης τάξης. Το δίκτυο εκπαιδεύτηκε για 300000 εποχές με τον adamW με ρυθμό μάθησης 10^{-4} και weight_decay 10^{-5} και κατόπιν για επιπλέον 15000 εποχές με L-BFGS.

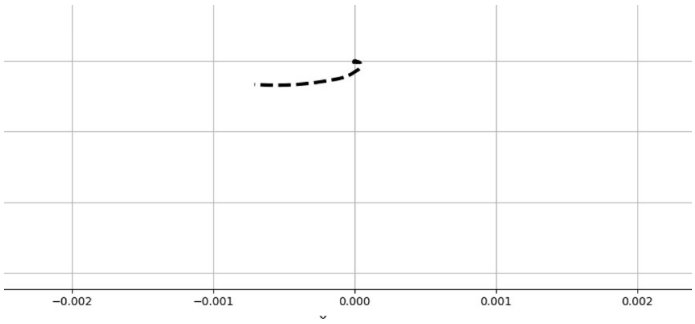


Σχήμα 47: Τροχιές Euler -Σύστημα 2ης τάξης: Απώλειες δίκτυο 3x64-adamW

Στο σχήμα απωλειών παρατηρούμε μια αργή σύγκλιση και την μεγάλη μείωση των απωλειών μετά την εκπαίδευση με τον αλγόριθμο L-BFGS.



Σχήμα 48: Τροχιές Euler -Σύστημα 2ης τάξης: Τροχιές δίκτυο 3x64



Σχήμα 49: Τροχιές Euler -Σύστημα 2ης τάξης: Λεπτομέρεια τροχιάς "ακίνητου" σώματος δίκτυο 3x64

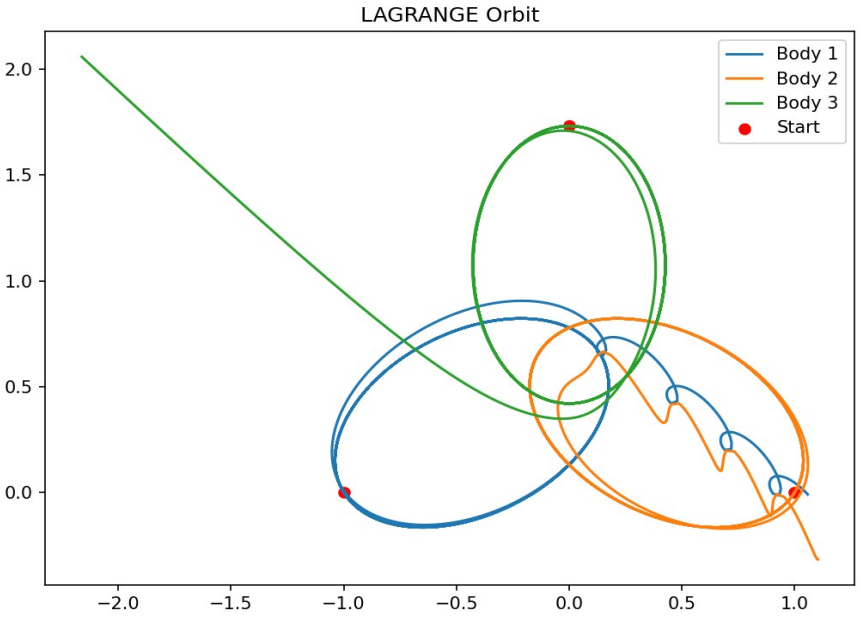
Στα σχήματα 48 και 49 παρατηρούμε επίσης καλά αποτελέσματα με το μικρότερο δίκτυο. Στο δίκτυο 3x64 ο αλγόριθμος L-BFGS χρειάσθηκε 3459 sec ενώ στο δίκτυο 3x128 5278 sec σε έναν επεξεργαστή intel i7. Η σωστή επιλογή του δικτύου είναι σημαντική ειδικά όταν χρησιμοποιούμε αλγόριθμους 2ης τάξης που είναι ιδιαίτερα απαιτητικοί σε χρόνο και μνήμη. Θα πρέπει να σημειωθεί ότι ο κρίσιμος παράγοντας εδώ δεν είναι τόσο η ισχύς του επεξεργαστή αλλά η μνήμη Ram (ή Vram στις GPU) που μπορεί να δεσμεύσει για την γρήγορη εκτέλεση των πράξεων. Στα συγκεκριμένα παραδείγματα δεσμεύτηκαν 0.5 GB μνήμης που θεωρείται οριακή.

4.2.5 Περιοδικές τροχιές Lagrange σύστημα 2ης τάξης - αλγόριθμος RAR

Για τις περιοδικές τροχιές Lagrange χρησιμοποιήθηκε μια συμμετρική διαμόρφωση με σώματα ίσης μάζας $m_1=m_2=m_3=1$ με αρχικές θέσεις στις κορυφές ενός ισόπλευρου τριγώνου πλευράς 2. Οι αρχικές ταχύτητες του συστήματος είναι τέτοιες ώστε το κέντρο μάζας του συστήματος να είναι διαρκώς ακίνητο. Τη αρχική χρονική στιγμή το σύστημα έχει μια γωνιακή ταχύτητα περιστροφής $\omega=0.3$ γύρω από το ακίνητο κέντρο μάζας. Οι αρχικές ταχύτητες έχουν μέτρο $u=\omega r$ και κατευθύνσεις κάθετες στις ακτίνες των σωμάτων από το κέντρο μάζας.

Η περίοδος του συστήματος μετρήθηκε με την βοήθεια του scipy intergator και βρέθηκε περίπου $T\approx 4.4$ κοντά στον χαρακτηριστικό χρόνο του συστήματος $T=2\pi\sqrt{\frac{\alpha^3}{G(m_1+m_2+m_3)}}$, όπου α ο μεγάλος ημιάξονας της ελλειπτικής τροχιάς. (Janssens, 2011)

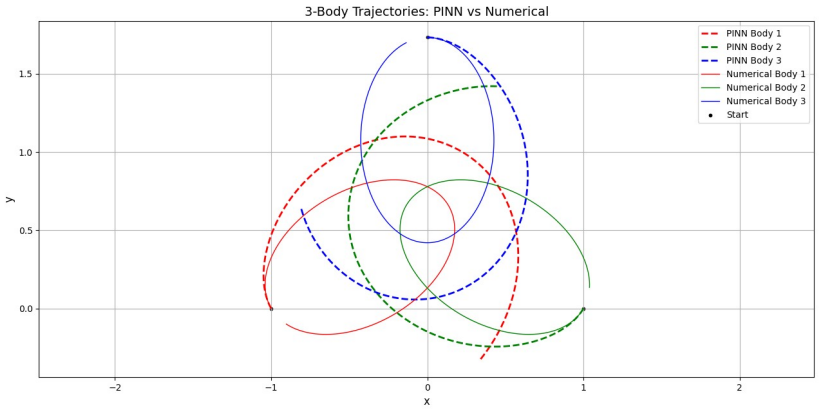
Επίσης όσον αφορά την ευστάθεια του συστήματος θα πρέπει να αναφέρουμε ότι με την παραπάνω διαμόρφωση είναι ασταθές. Επομένως μικρά σφάλματα στις τροχιές μεγεθύνονται με την πάροδο του χρόνου. Αυτό μπορεί να επιβεβαιωθεί και με την προσομοίωση με τον scipy intergator όπου η λύση αποκλίνει μετά από περίπου 7 περιόδους όπως φαίνεται στο παρακάτω σχήμα.



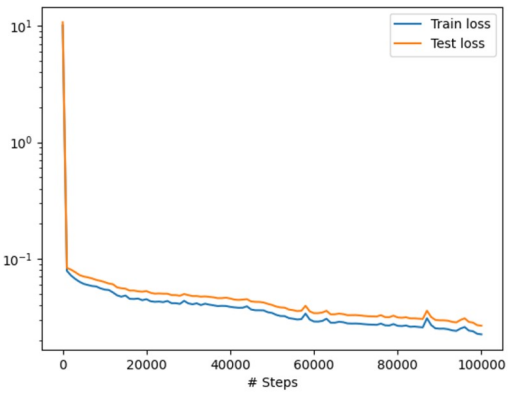
Σχήμα 50: Αστάθεια τροχιών Lagrange

Για περισσότερα σχετικά με την ευστάθεια των τροχιών Lagrange και τα κριτήρια ευστάθειας υπάρχουν διάφορες μελέτες. (Roberts, 2002)

Η αρχική μας διαμόρφωση δεν είναι σε σωστή κλίμακα για την εκπαίδευση ενός pinn. Ο λόγος είναι ότι ο χρόνος εκπαίδευσης θα πρέπει να είναι τουλάχιστον μία περίοδος που σημαίνει εκπαίδευση για $t \approx 4$. Όμως για την σωστή σύγκλιση του pinn χρειαζόμαστε εισόδους που να βρίσκονται στην γραμμική περιοχή της υπερβολικής εφαπτομένης που είναι η συνάρτηση ενεργοποίησης που χρησιμοποιούμε. Τιμές εκτός της περιοχής $[-2,2]$ βρίσκονται στην περιοχή κορεσμού της συνάρτησης και κάνουν την εκπαίδευση του δικτύου δύσκολη και σε κάποιες περιπτώσεις αδύνατη. Για να διαπιστώσουμε το πρόβλημα εκπαιδεύσαμε το pinn στην αρχική κλίμακα χρόνου. Χρησιμοποιήθηκε ένα δίκτυο 3 στρωμάτων με 64 νευρώνες ανά στρώμα , ρυθμό μάθησης 10^{-4} ,ίσα βάρη και adam optimizer. Επίσης επιβάλαμε τις αρχικές συνθήκες με hard contrains. Μετά από 100000 εποχές εκπαίδευσης πήραμε τα παρακάτω αποτελέσματα:



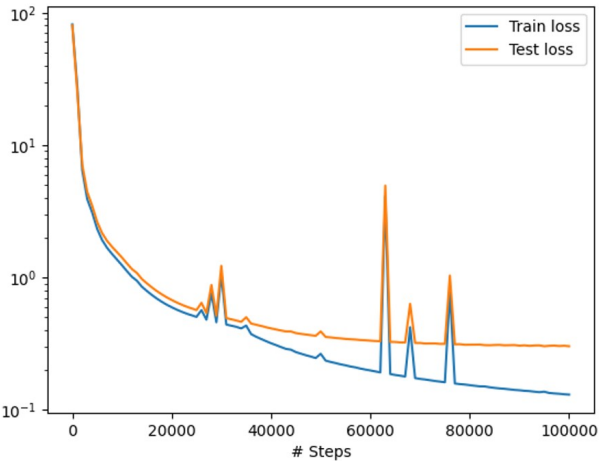
Σχήμα 51: Τροχιά Lagrange - pinn με εσφαλμένη κλίμακα



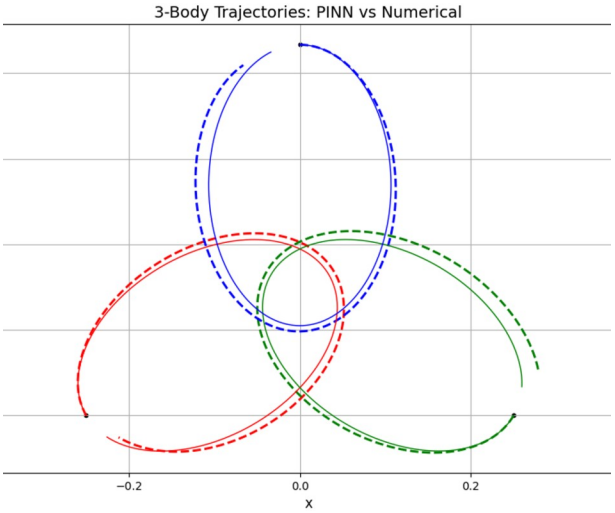
Σχήμα 52: Απώλειες - Τροχιά Lagrange - pinn με εσφαλμένη κλίμακα

Παρατηρούμε ότι μετά από μεγάλο αριθμό εποχών έχουμε μικρή ταύτιση με την αριθμητική λύση. Επίσης αυξάνοντας τις εποχές εκπαίδευσης παρατηρούμε μείωση των απωλειών χωρίς σημαντική βελτίωση των τροχιών.

Κατόπιν αλλάξαμε τις κλίμακες χρόνου και χώρου κάνοντας έναν μετασχηματισμό που αφήνει τις αρχικές εξισώσεις του συστήματος αναλλοίωτες (παράγραφος 4.1.2) έτσι ώστε ο χρόνος μιας περιόδου να είναι $t \approx 0.5$ ώστε να έχουμε και περιθώριο για εκπαίδευση και σε δύο περιόδους. Χρησιμοποιήθηκαν οι ίδιες υπερπαράμετροι – εποχές εκπαίδευσης με το προηγούμενο τεστ ώστε να υπάρχει άμεση σύγκριση. Παρακάτω φαίνονται τα αποτελέσματα της εκπαίδευσης.



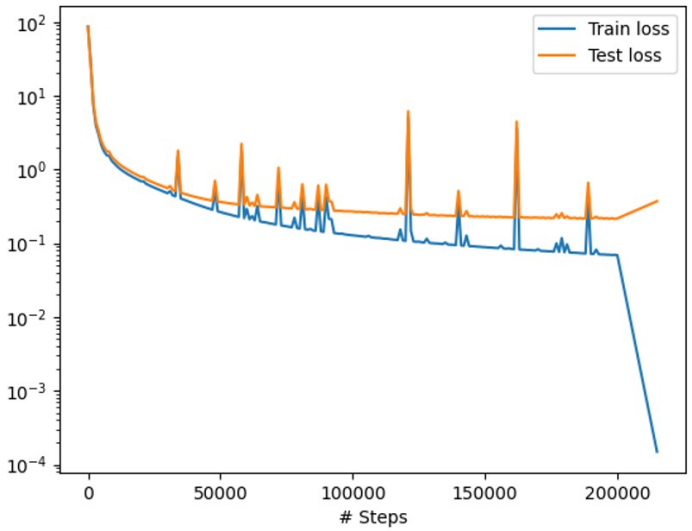
Σχήμα 54: Απώλειες - Τροχιά Lagrange - pinn μετά την αλλαγή κλίμακας



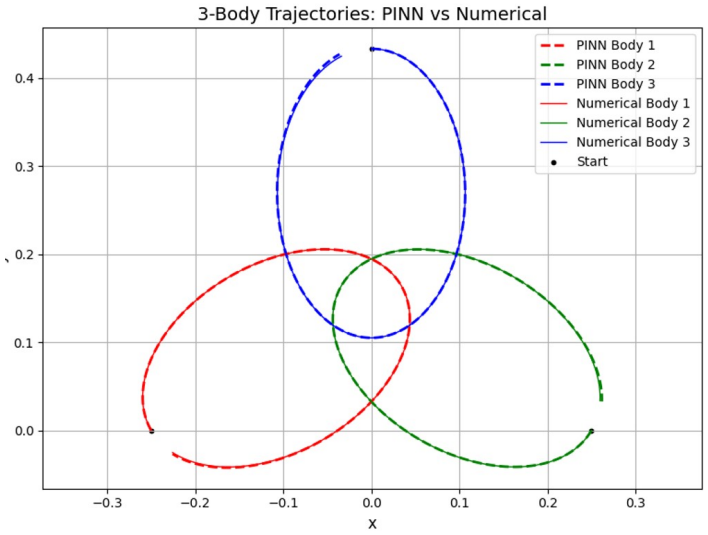
Σχήμα 53: Τροχιά Lagrange - pinn μετά την αλλαγή κλίμακας

Παρατηρούμε ότι μετά την αλλαγή κλίμακας έχουμε πολύ καλύτερα αποτελέσματα.

Στο επόμενο πείραμα εκπαιδεύσαμε ένα δίκτυο 3x64 με ίσα βάρη αρχικά για 200000 εποχές με τον αλγόριθμο adam και με ρυθμό μάθησης 10^{-4} και στην συνέχεια με τον αλγόριθμο L-BFGS για 15000 εποχές. Οι αρχικές συνθήκες επιβλήθηκαν με hard constrains.



Σχήμα 55: Απώλειες-Τροχιές Lagrange - L-BFGS



Σχήμα 56: Τροχιές Lagrange - L-BFGS

Στο σχήμα 55 παρατηρούμε μια εντυπωσιακή βελτίωση των απωλειών και στο σχήμα 56 σχεδόν πλήρη ταύτιση με την αριθμητική λύση. Όμως από το διάγραμμα απωλειών φαίνεται ότι οι

απώλειες κατά την εκπαίδευση (train-loss) απέχουν σχεδόν 5 τάξεις μεγέθους από τις απώλειες σε άλλα τυχαία σημεία στα οποία δεν εκπαιδεύτηκε το δίκτυο (test-loss). Αυτό σημαίνει ότι κατά την εκπαίδευση με τον αλγόριθμο L-BFGS παρά την εντυπωσιακή μείωση των απωλειών το δίκτυο δεν γενίκευσε. Ο λόγος είναι ότι ο αλγόριθμος δεύτερης τάξης πρέπει να χρησιμοποιείται μετά από ικανή εκπαίδευση του δικτύου. Όμως το διάγραμμα απωλειών δείχνει ότι η αρχική εκπαίδευση με το αλγόριθμο adam για 200000 εποχές δεν είναι ικανοποιητική και πιθανόν χρειάζονται περισσότερα σημεία εκπαίδευσης (collocation points).

Για την καλύτερη εκπαίδευση του δικτύου χρησιμοποιήθηκε η τεχνική **RAR** (Residual-based adaptive refinement) που προτάθηκε για τα pinns από τους *Lu, L., Meng, X., Mao, Z., & Karniadakis,G.E.(2021)*.(Lu κ.ά., 2021)

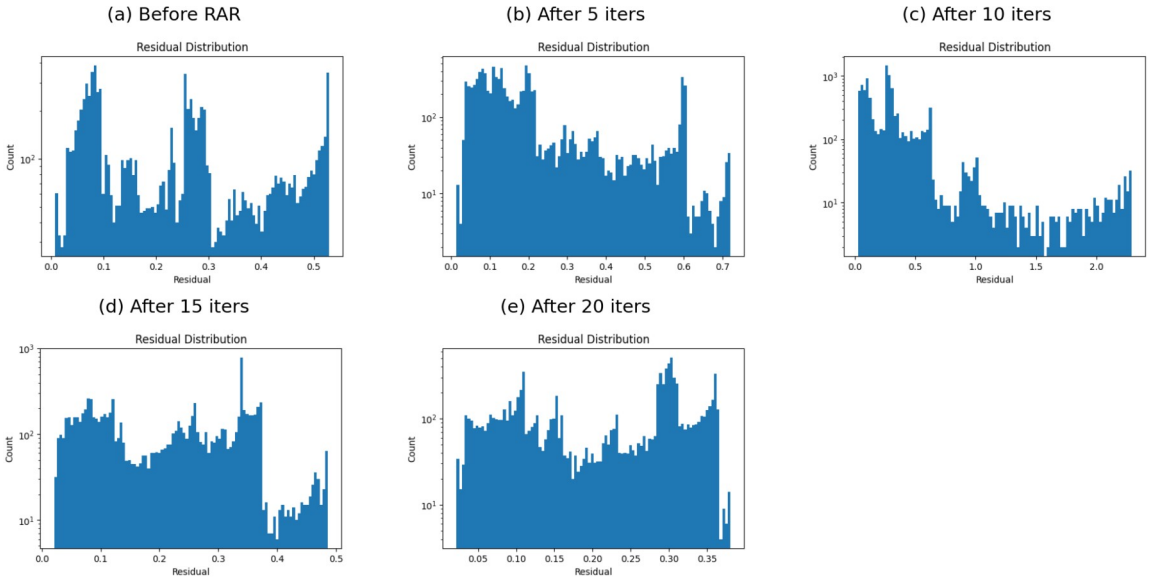
Ένα παράδειγμα μπορεί να βρεθεί στην τεκμηρίωση της βιβλιοθήκης deepxde Burgers equation with RAR

Τα βήματα του αλγόριθμου του χρησιμοποιήσαμε είναι τα παρακάτω:

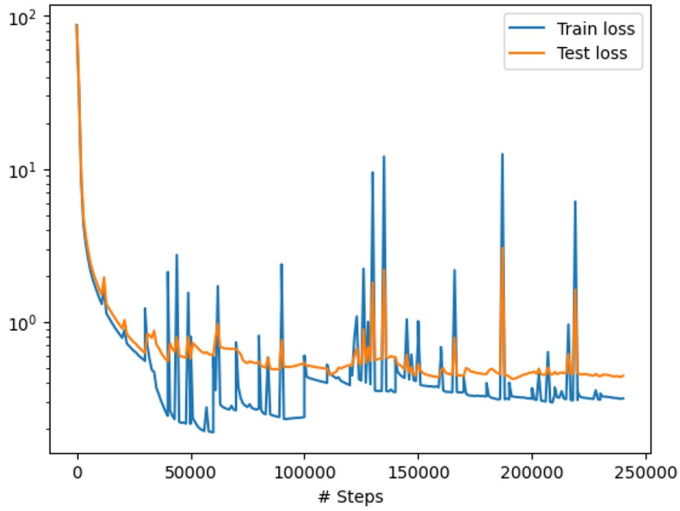
- Επιλογή ενός τυχαίου δείγματος Δ εισόδων (χρονικών στιγμών)
- Επανάληψη N φορές (ή μέχρι να ικανοποιηθεί μια συνθήκη)
- Υπολογισμός των υπολοίπων του τυχαίου δείγματος
- Για κάθε σημείο του δείγματος (χρονική στιγμή) υπολογισμός του μέσου όρου της απόλυτης τιμής των υπολοίπων. Στην συγκεκριμένη περίπτωση έχουμε έξι υπόλοιπα.
- Επιλογή των χρονικών στιγμών με τα Σ μεγαλύτερα υπόλοιπα
- Πρόσθεση στο σύνολο σημείων εκπαίδευσης των Σ σημείων
- Εκπαίδευση του δικτύου για Ε εποχές εκπαίδευσης

Ο κώδικας για τον αλγόριθμο RAR βρίσκεται στο *Παράρτημα Δ*.

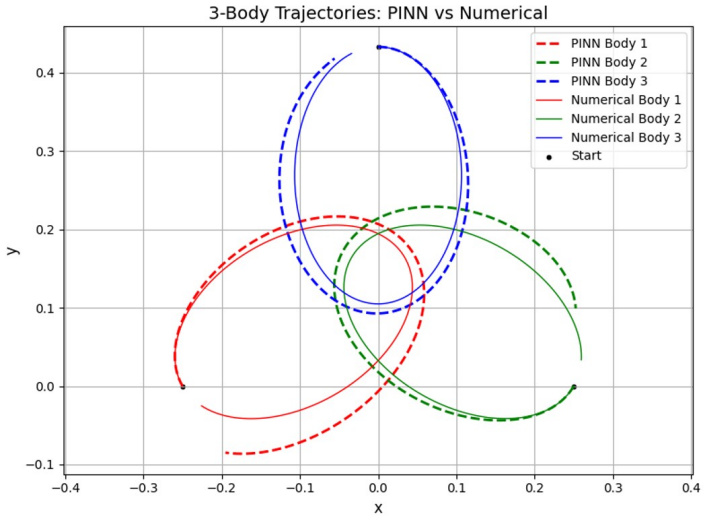
Χρησιμοποιώντας τις ίδιες παραμέτρους και τον ίδιο αριθμό σημείων εκπαίδευσης (64) εκπαιδεύσαμε αρχικά το δίκτυο για 30000 εποχές με αλγόριθμο adam. Στην συνέχεια εκπαιδεύσαμε με τον αλγόριθμο RAR με 21 επαναλήψεις και εκπαίδευση για 10000 εποχές για κάθε επανάληψη με αλγόριθμο adam. Σε κάθε επανάληψη του αλγόριθμου RAR προσθέτουμε 64 νέα σημεία εκπαίδευσης που έχουν επιλεχθεί από ένα αρχικό τυχαίο δείγμα 10000 σημείων. Παρακάτω φαίνονται τα ιστογράμματα με την κατανομή των υπολοίπων στα τυχαία σημεία κατά την διάρκεια της εκπαίδευσης.



Σχήμα 57: Τροχιές Lagrange: Κατανομή υπολοίπων με αλγόριθμο RAR

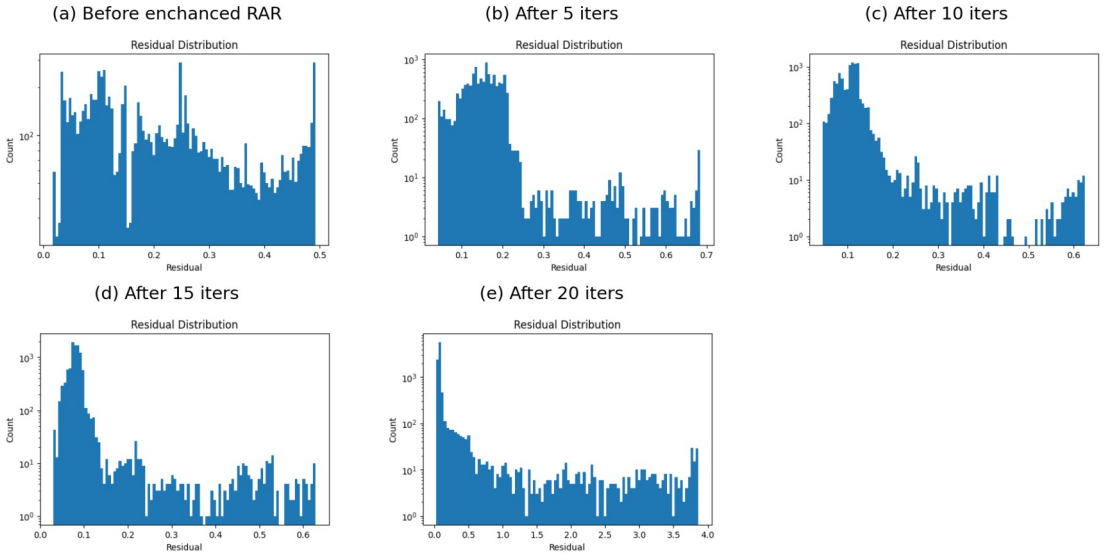


Σχήμα 59: Τροχιές Lagrange: Απώλειες με αλγόριθμο RAR



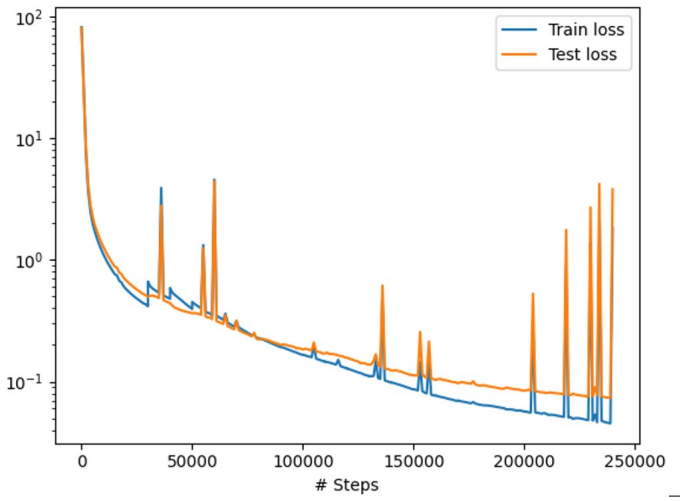
Σχήμα 58: Τροχιές Lagrange: Τροχιές με αλγόριθμο RAR

Από το διάγραμμα ολικών απωλειών φαίνεται ότι το train-loss και το test-loss πλησιάζουν όμως η εκπαίδευση του δικτύου δεν βελτιώθηκε με τον αλγόριθμο RAR όπως φαίνεται και στο σχήμα των τροχιών. Επομένως απαιτούνται περισσότερα σημεία. Όμως η εκπαίδευση με περισσότερα σημεία είναι χρονοβόρα ειδικά για το δευτεροβάθμιο σύστημα. Για την αντιμετώπιση του προβλήματος χρησιμοποιήσαμε μια παραλλαγή του αλγόριθμου όπου αντί να προσθέτουμε τα 64 σημεία με τις μεγαλύτερες απώλειες επιλέξαμε την περιοχή μεταξύ του 60% με 95% του δείγματος με τα σημεία υψηλών απωλειών και από αυτό το δείγμα επιλέξαμε τυχαία τα 64 σημεία που προσθέτουμε σε κάθε επανάληψη (*Παράρτημα Ε*). Παρακάτω φαίνονται τα ιστογράμματα με την κατανομή των υπολοίπων καθώς και οι ολικές απώλειες και τροχιές που προέκυψαν.

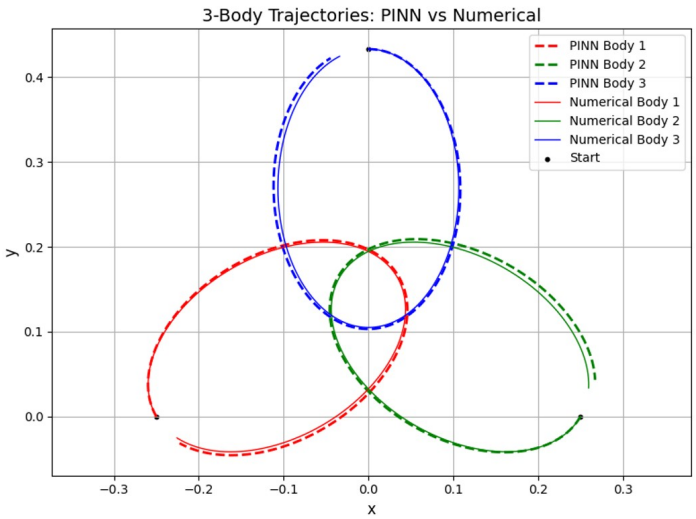


Σχήμα 60: Τροχιές Lagrange: Κατανομή υπολοίπων με παραλλαγή αλγόριθμου RAR

Από τα διαγράμματα με τις κατανομές παρατηρούμε μία σταθερή βελτίωση των υπολοίπων.

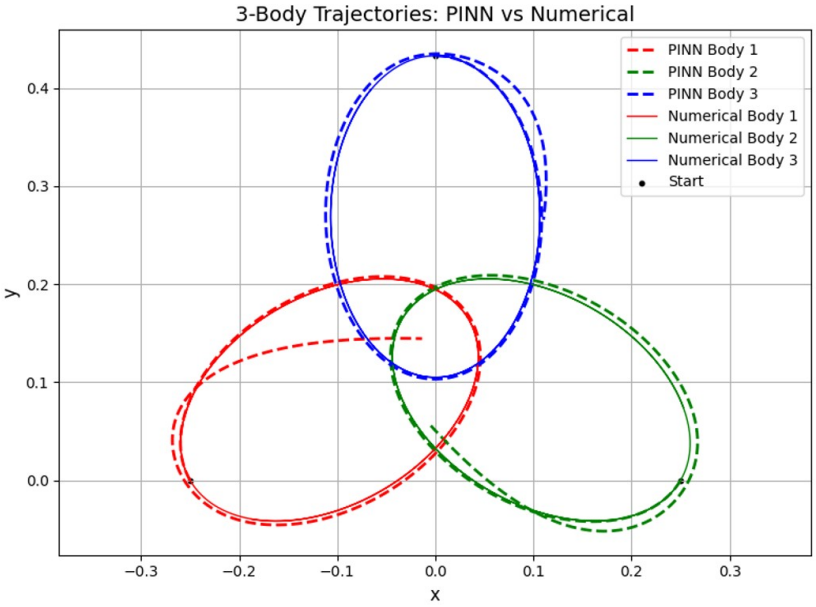


Σχήμα 62: Τροχιές Lagrange: Απώλειες με παραλλαγή αλγόριθμου RAR



Σχήμα 61: Τροχιές Lagrange:Τροχιές με παραλλαγή αλγόριθμου RAR

Από το σχήμα απωλειών παρατηρούμε μια σταθερή σύγκλιση τόσο στα σημεία εκπαίδευσης (train-loss) όσο και στα τυχαία σημεία ελέγχου (test-loss). Επίσης οι τροχιές πλησιάζουν ικανοποιητικά την αριθμητική λύση. Το pinn μας δίνει και μία πρόβλεψη (extrapolation) όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 63: Τροχιές Lagrange: Extrapolation για χρόνο διπλάσιο της εκπαίδευσης-παραλλαγή αλγόριθμου RAR

Στο σχήμα 63 παρατηρούμε την απόκριση του pinn για χρόνο διπλάσιο από τον χρόνο εκπαίδευσης. Το pinn συμπληρώνει σωστά την τροχιά για κάποιο κλάσμα της περιόδου και κατόπιν έχει σωστή κλίση τροχιών. Αυτό δείχνει ότι ο παραλλαγμένος αλγόριθμος RAR βοηθά το δίκτυο να γενικεύσει.

Τα πειράματα με τον αλγόριθμο RAR εκτελέστηκαν στην TPUv6e1 του google colab λόγω του αρκετού χρόνου που χρειάζονται. Η TPU (tensor processing unit) είναι ειδικό hardware για τον αποτελεσματικό υπολογισμό πράξεων με πίνακες. Η εκτέλεση έγινε χωρίς βελτιστοποίηση (optimization) και παρατηρήθηκε ταχύτητα 3 με 4 φορές μεγαλύτερη από το hardware των υπόλοιπων πειραμάτων (intel i7- home computer).

4.2.6 Περιοδικές τροχιές σχήματος οκτώ (figure eight) – σύστημα 2ης τάξης

Για τις περιοδικές τροχιές figure -8 χρησιμοποιήσαμε σώματα ίσης μάζας $m_1=m_2=m_3=1$.

Οι αρχικές συνθήκες για τις θέσεις και τις ταχύτητες ήταν:

$$y_0=[\ x_1(0)=-0.97000436, y_1(0)=0.24308753, x_2(0)=0.97000436, y_2(0)=-0.24308753, x_3(0)=0, y_3(0)=0]$$

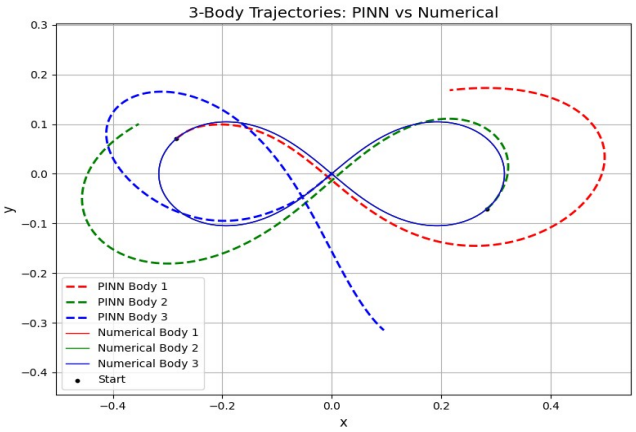
$$u_0=[\dot{x}_1(0)=0.4662036850, \dot{y}_1(0)=0.4323657300, \dot{x}_2(0)=0.4662036850, \dot{y}_2(0)=0.4323657300, \dot{x}_3(0)=-0.93240737, \dot{y}_3(0)=-0.86473146]$$

Αξίζει να παρατηρήσουμε ότι $x_1+x_2+x_3=0$, $y_1+y_2+y_3=0$, $v_{x1}+v_{x2}+v_{x3}=0$, $v_{y1}+v_{y2}+v_{y3}=0$. Με αυτές τις αρχικές συνθήκες τα τρία σώματα εκτελούν την ίδια τροχιά σχήματος 8 με διαφορά φάσης 120^0.

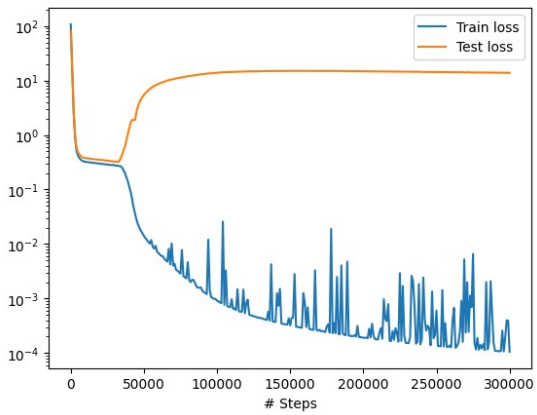
Οι παραπάνω τιμές δόθηκαν από τους Alain Chenciner και Richard Montgomery (Chenciner & Montgomery, 2000)

Η περίοδος των τροχιών είναι T=6.32591398. Έγινε κατάλληλη αλλαγή κλίμακας ώστε T=1 και ταυτόχρονα μία αλλαγή κλίμακας στο μήκος ώστε οι αρχικές εξισώσεις να παραμείνουν αναλλοίωτες. Ταυτόχρονα έγιναν και οι κατάλληλες αλλαγές στις αρχικές συνθήκες του προβλήματος. Επίσης οι αρχικές συνθήκες επιβλήθηκαν με έναν μετασχηματισμό εξόδου (hard constrains) όπως και στις προηγούμενες παραγράφους. Τα τεστ έγιναν στην TPUv6e1 (tensor processing unit) που προσφέρει το google colab.

Αρχικά επιλέξαμε ένα δίκτυο με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα, adam optimizer με ρυθμό μάθησης 10^{-4} , βάρη ίσα με 1 , και 300000 εποχές εκπαίδευσης.



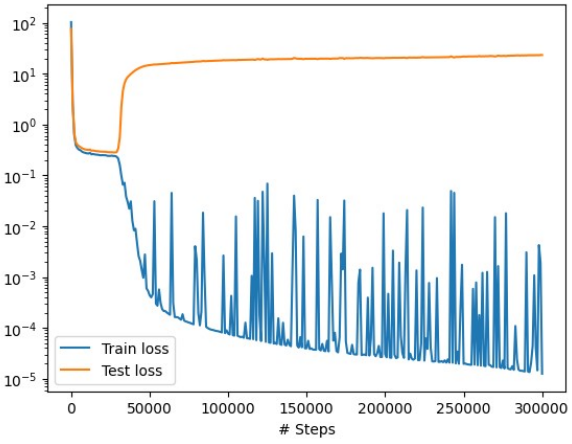
Σχήμα 64: Τροχιά figure - 8 δίκτυο 3x64



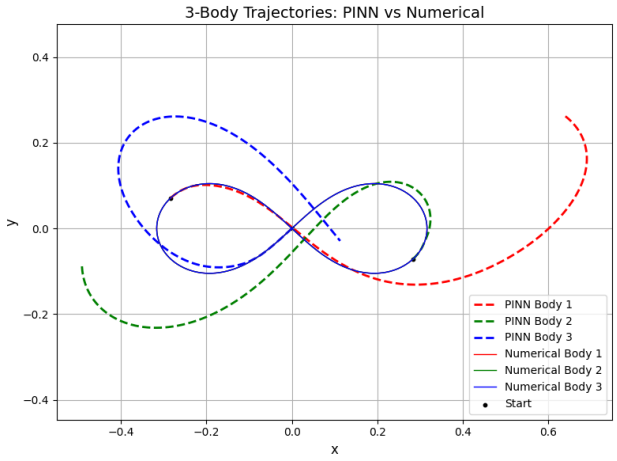
Σχήμα 65: Απώλειες figure-8 δίκτυο 3x64

Από το διάγραμμα απωλειών παρατηρούμε ότι ενώ το δίκτυο εκπαιδεύεται στο training set δεν γενικεύει αφού το test loss μετά από περίπου 35000 εποχές είναι πολύ μεγάλο. Επίσης οι τροχιές δεν είναι ακολουθούν την αριθμητική λύση. Το πρόβλημα είναι μη αναμενόμενο διότι η δυναμική στην τροχιά figure 8 δεν παρουσιάζει μεγάλες δυσκολίες. Τα σώματα βρίσκονται διαρκώς σε κάποια απόσταση και δεν πλησιάζουν ποτέ αρκετά κοντά όπως στις τροχιές Euler και Lagrange. Επίσης στο σύστημα 2ης τάξης που χρησιμοποιήσαμε έχουμε μόνο 6 όρους σφαλμάτων που είναι αρκετά ισορροπημένοι. Φαίνεται ότι το δίκτυο δυσκολεύεται να μας δώσει τρεις ίδιες τροχιές με διαφορά φάσης 120º. Επομένως εδώ έχουμε ένα διαφορετικό πρόβλημα σε σχέση με το παράδειγμα των τροχιών Lagrange που είδαμε στην προηγούμενη παράγραφο. Το δίκτυο δεν είναι αρκετά βαθύ ώστε να μας δώσει αυτές τις ιδιαίτερα συμμετρικές τροχιές. Επίσης μια άλλη παρατήρηση που κάναμε με το δίκτυο 3x64 είναι ότι μικρές αλλαγές βαρών επηρεάζουν σημαντικά την εξέλιξη της εκπαίδευσης του δικτύου.

Το επόμενο πείραμα έγινε με ένα δίκτυο με 5 κρυφά στρώματα και 64 νευρώνες ανά στρώμα με τις ίδιες παραμέτρους.



Σχήμα 67: Απώλειες figure - 8 δίκτυο 5x64



Σχήμα 66: Τροχιά figure - 8 δίκτυο 5x64

Από το σχήμα απωλειών παρατηρούμε ότι δεν έχουμε κάποια αλλαγή εκτός από το ότι τα σημεία εκπαίδευσης δίνουν ακόμα μικρότερες απώλειες όμως από το σημεία ελέγχου (test-loss) παρατηρούμε το ίδιο πρόβλημα όπως στο δίκτυο 3x64. Στις τροχιές επίσης δεν έχουμε κάποια βελτίωση.

Ένας τρόπος να αντιμετωπισθεί το πρόβλημα είναι να εισάγουμε έναν μετασχηματισμό στην είσοδο του δικτύου τύπου Fourier (Fourier feature transform).Ένα παράδειγμα από την βιβλιοθήκη deepxde μπορεί να βρεθεί στον σύνδεσμο Lotka-Voletrra equation

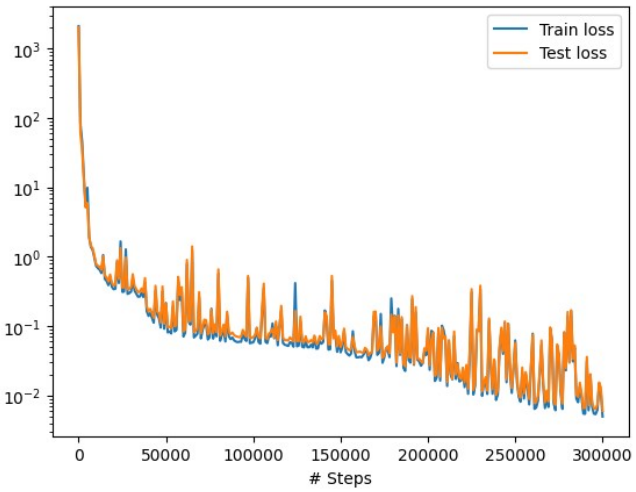
Εφόσον γνωρίζουμε ότι το πρόβλημα είναι περιοδικό με περίοδο T=1 και επίσης ότι έχουμε τρεις ίδιες λύσεις με διαφορά φάσης $2\pi/3$ εισάγουμε στην είσοδο τον μετ/μο:

$$\Phi(t)=[t,\sin(\omega t),\cos(\omega t),\sin(\omega t+\frac{2\pi}{3}),\cos(\omega t+\frac{2\pi}{3}),\sin(2\omega t+\frac{2\pi}{3}),\cos(2\omega t+\frac{2\pi}{3})]$$

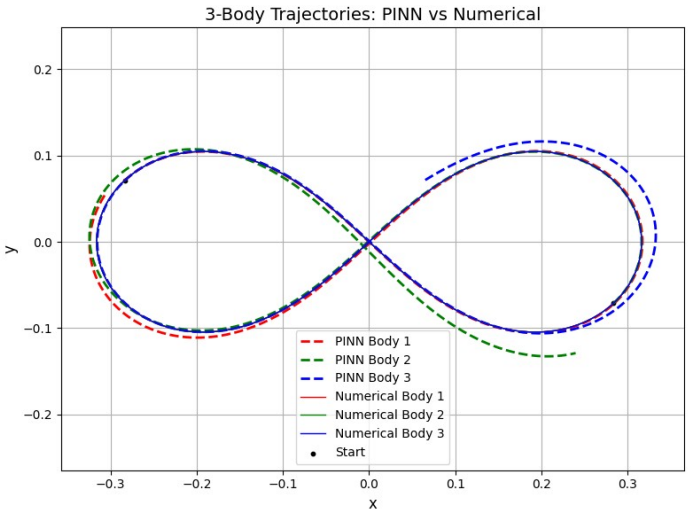
με βασική συχνότητα $\omega=\frac{2\pi}{T}$.

Επομένως αντί για το δίκτυο $N(t,\theta)$ έχουμε το δίκτυο $N(\Phi(t),\theta)$.

Χρησιμοποιήθηκε το δίκτυο 5x64 με τις ίδιες παραμέτρους και τον παραπάνω μετ/μο.



Σχήμα 69: Απώλειες figure 8 με μετ/μο εισόδου

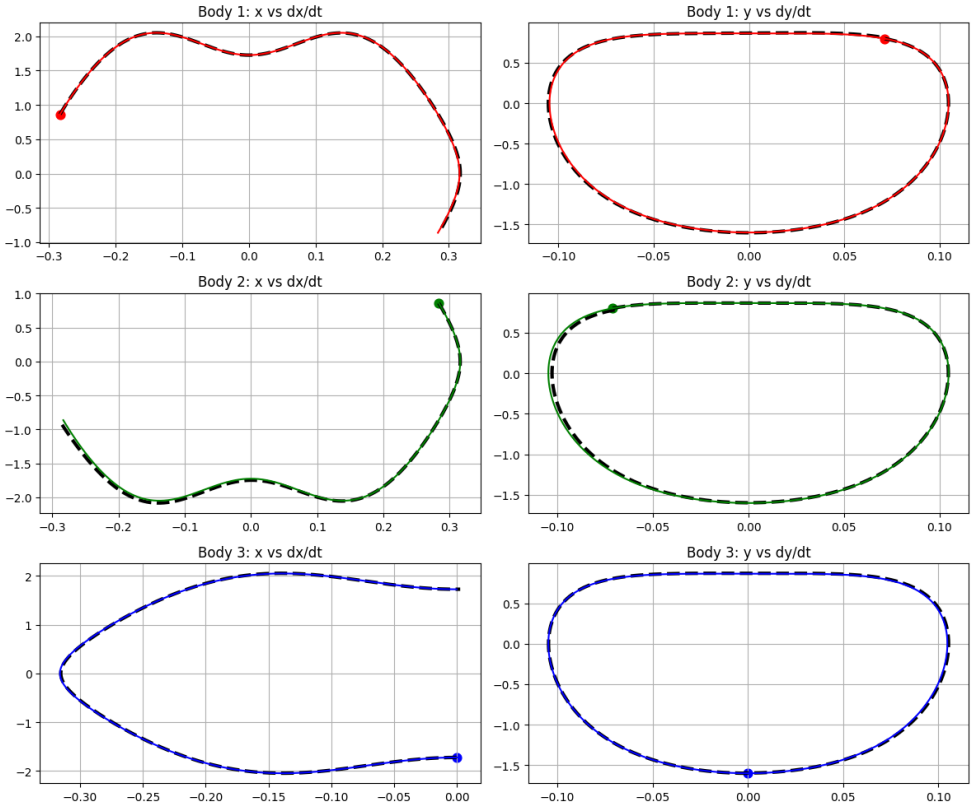


Σχήμα 68: Τροχιές figure 8 με μετ/μο εισόδου

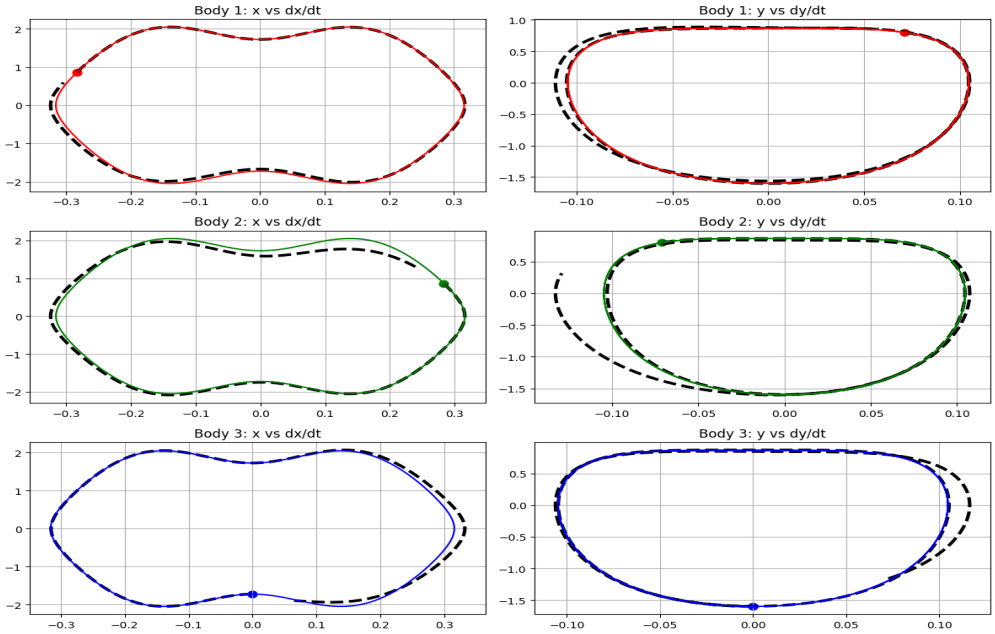
Απο τα παραπάνω σχήματα παρατηρούμε μία σταθερή σύγκλιση όπου train-loss και test-loss συμβαδίζουν. Επίσης πολύ καλύτερη προσέγγιση των τροχιών με την αριθμητική λύση.

Για καλύτερη εποπτεία δίνουμε και τα φασικά διαγράμματα για χρόνο t=T/2 και t=T. Ο λόγος που δίνουμε τα διαγράμματα για μια ημιπερίοδο και κατόπιν για μία περίοδο είναι η ευκρίνεια. Ενώ τα φασικά διαγράμματα για τις x - συντεταγμένες εκτελούν μία επανάληψη ανά περίοδο τα αντίστοιχα διαγράμματα για τις y - συντεταγμένες εκτελούν δύο περιστροφές ανά περίοδο λόγω του σχήματος 8 της τροχιάς.

Ο κώδικας που χρησιμοποιήθηκε για τον υπολογισμό και την εμφάνιση των φασικών διαγραμμάτων φαίνεται στο *παράρτημα Z*.



Σχήμα 70: Figure 8 - Φασικά διαγράμματα για μία ημιπερίοδο



Σχήμα 71: Figure 8 - Φασικά διαγράμματα για μία περίοδο

Από τα φασικά διαγράμματα παρατηρούμε καλή ταύτιση για την ημιπερίοδο όμως σε χρόνο μίας περιόδου ειδικά τα διαγράμματα με τις y – συνιστώσες αρχίζουν να αποκλίνουν από την αριθμητική λύση. Το σώμα 1 έχει την καλύτερη συμπεριφορά και ακολουθούν τα σώματα 3 και 2.

Επίσης θα πρέπει να αναφέρουμε ότι αν εφαρμόσουμε έναν μετ/μο εισόδου χωρίς να λάβουμε υπ όψιν τις διαφορές φάσεις π.χ $\Phi(t)=[t,\sin(\omega t),\cos(\omega t),\sin(2\omega t),\cos(2\omega t)]$ τότε έχουμε επίσης καλά αποτελέσματα άλλα ο μετ/μος με τις διαφορές φάσης έχει λίγο καλύτερα αποτελέσματα.

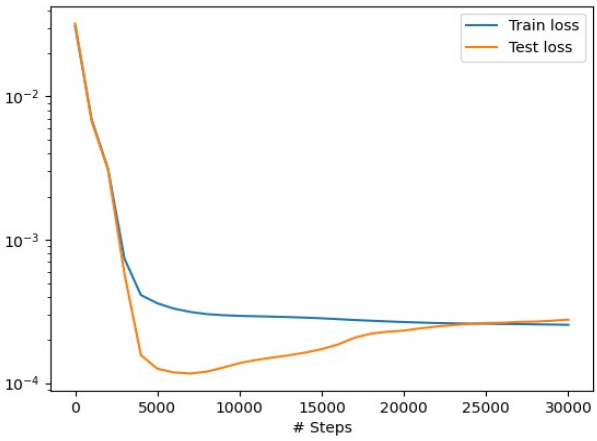
4.3 Εκπαίδευση με εξωτερικά δεδομένα

4.3.1 Σύγκριση απλού δικτύου με Pinn ενημερωμένο με αρχικές συνθήκες.

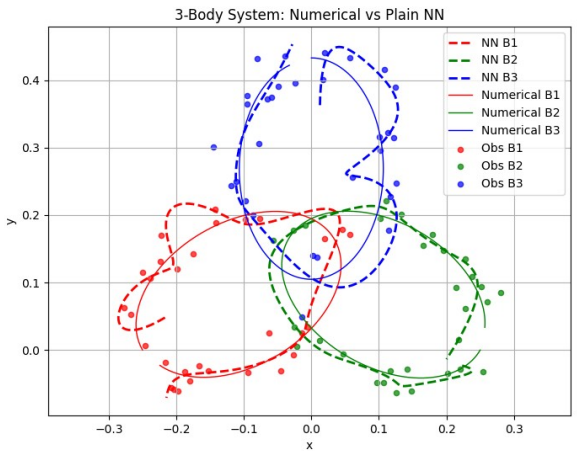
Στην παρούσα παράγραφο θα συγκρίνουμε ένα απλό FeedForward νευρωνικό δίκτυο με ένα Pinn που έχουμε επιβάλει τις αρχικές συνθήκες του προβλήματος. Και τα δύο δίκτυα θα εκπαιδευτούν με τα ίδια εξωτερικά δεδομένα που περιέχουν μεγάλο επίπεδο θορύβου (20%). Επίσης θέσαμε τα ίδια seed στις μηχανές τυχαίων αριθμών ώστε να έχουμε όσο το δυνατόν παρόμοιες αρχικές διαμορφώσεις των βαρών των δικτύων καθώς και επαναληψιμότητα των πειραμάτων.

Χρησιμοποιήθηκαν δίκτυα με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα και adam με ρυθμό μάθησης 10^{-4}. Και τα δύο δίκτυα εκπαιδεύτηκαν με *90 δεδομένα με θόρυβο* για τις τροχιές Lagrange που έχουμε ήδη μελετήσει. Έγινε εκπαίδευση για 30000 και 120000 εποχές.

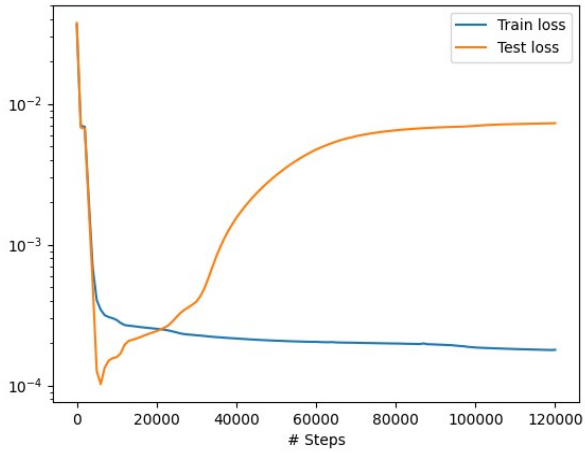
Παρακάτω δίνουμε τα διαγράμματα από την εκπαίδευση του απλού νευρωνικού δικτύου.



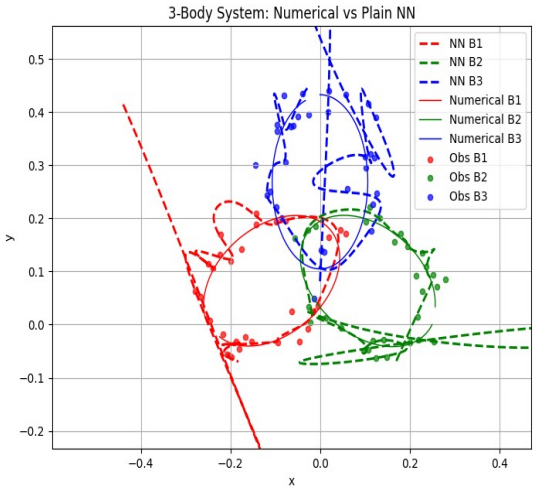
Σχήμα 73: Απώλειες - Απλό νευρωνικό δίκτυο - εξωτερικά δεδομένα - 30000 εποχές



Σχήμα 72: Τροχιές - Απλό νευρωνικό δίκτυο - εξωτερικά δεδομένα - 30000 εποχές



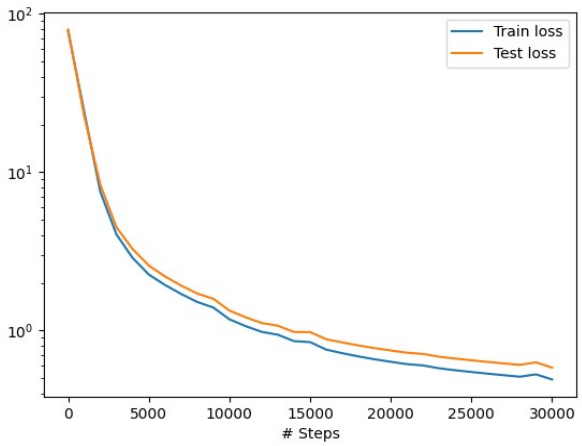
Σχήμα 75: Απώλειες - Απλό νευρωνικό δίκτυο - εξωτερικά δεδομένα -120000 εποχές



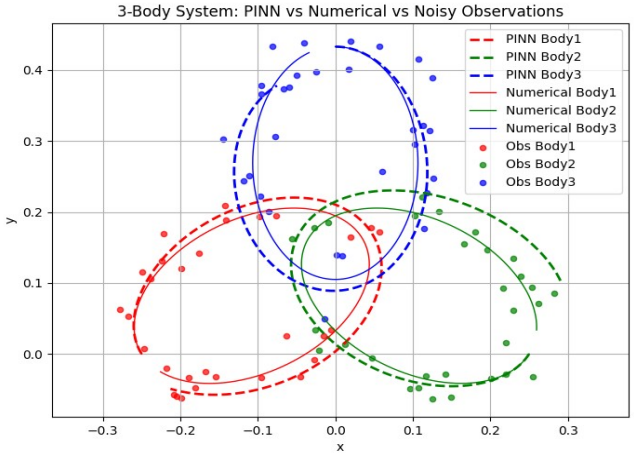
Σχήμα 74: Τροχιές - Απλό νευρωνικό δίκτυο - εξωτερικά δεδομένα - 120000 εποχές

Από τα παραπάνω διαγράμματα φαίνεται ότι το απλό νευρωνικό δίκτυο δεν μπορεί να προσεγγίσει ομαλά την τροχιά. Οι διακεκομμένες γραμμές που είναι η πρόβλεψη του δικτύου προσπαθούν να περιλάβουν όλα τα σημεία δίνοντας παραμορφωμένες τροχιές. Επίσης η περαιτέρω εκπαίδευση δίνει χειρότερα αποτελέσματα.

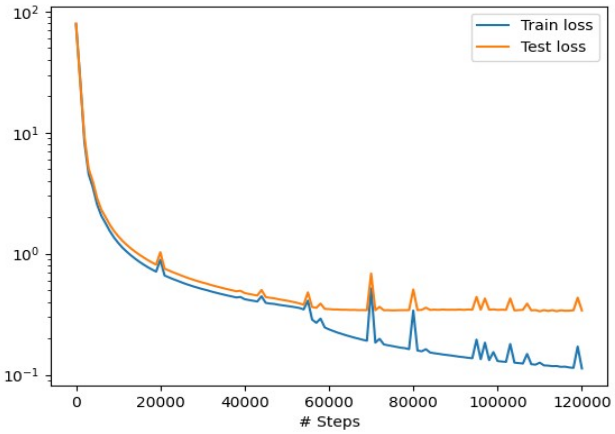
Παρακάτω δίνουμε τα αντίστοιχα διαγράμματα από την εκπαίδευση του Pinn.



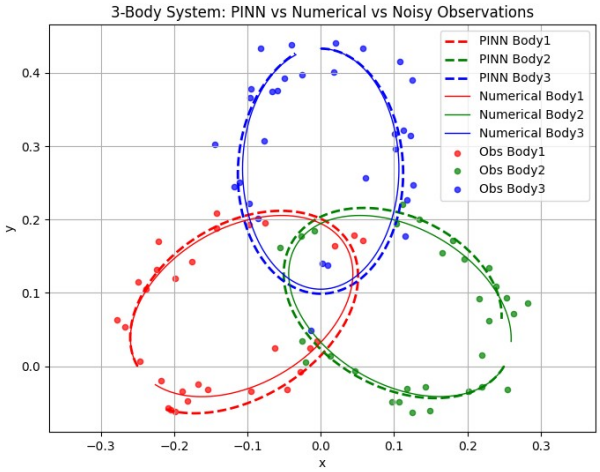
Σχήμα 77: Απώλειες - Pinn - εξωτερικά δεδομένα -30000 εποχές



Σχήμα 76: Τροχιές - Pinn - εξωτερικά δεδομένα - 30000 εποχές



Σχήμα 79: Απώλειες - Pinn - εξωτερικά δεδομένα -120000 εποχές



Σχήμα 78: Τροχιές - Pinn - εξωτερικά δεδομένα - 120000 εποχές

Από τα διαγράμματα φαίνεται καθαρά ότι το pinn έμαθε τις σωστές ομαλές τροχιές παρά το μεγάλο επίπεδο θορύβου. Επίσης οι τροχιές που δίνει το pinn βελτιώνονται σημαντικά με παραπάνω εποχές εκπαίδευσης.

Το συμπέρασμα είναι ότι τα pinns μπορούν να δώσουν πολύ καλύτερα αποτελέσματα ειδικά όταν έχουμε λίγα δεδομένα με υψηλά επίπεδα θορύβου.

Εδώ θα μπορούσε να ασκηθεί η κριτική ότι το pinn γνώριζε τις αρχικές συνθήκες και ίσως η παραπάνω σύγκριση δεν είναι αρκετά δίκαιη. Στην συγκεκριμένη περίπτωση των τροχιών Lagrange είχαμε μια αριθμητική λύση όμως υπάρχουν συστήματα ιδιαίτερα πολύπλοκα που παρόλο που γνωρίζουμε τις αρχικές συνθήκες δεν μπορούμε εύκολα να υπολογίσουμε αριθμητικά την εξέλιξή τους. Τέτοια παραδείγματα είναι πολλά π.χ τα καιρικά φαινόμενα , οι σεισμοί , οι εξέλιξη των τιμών μιας μετοχής κ.τ.λ. Σε αυτές τις περιπτώσεις γνωρίζουμε κάποιους νόμους που διέπουν τα φαινόμενα άλλα η γνώση μας δεν είναι αρκετή για να προβλέψουμε την εξέλιξή τους ή έχουμε τυχαιότητα που δεν μπορεί να μοντελοποιηθεί. Επομένως σε πολύπλοκα προβλήματα που έχουμε ελλειπή γνώση και λίγα πειραματικά δεδομένα τα pinns μπορούν να δώσουν χρήσιμες πληροφορίες για το σύστημα.

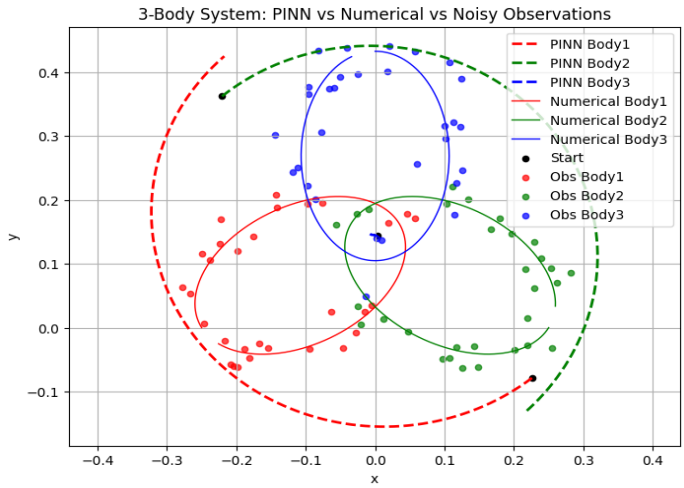
Στην επόμενη παράγραφο θα διερευνήσουμε την συμπεριφορά και τις δυνατότητες ενός pinn χωρίς καμία γνώση των αρχικών συνθηκών.

4.3.2 Εκπαίδευση Pinn με εξωτερικά δεδομένα χωρίς γνώση των αρχικών συνθηκών

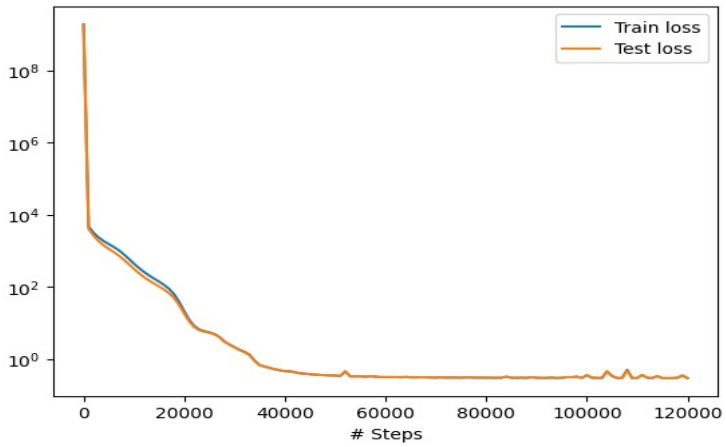
Σε αυτήν την παράγραφο θα εκπαιδεύσουμε το Pinn με τα εξωτερικά δεδομένα που χρησιμοποιήσαμε στην προηγούμενη παράγραφο *χωρίς όμως καμία μορφή αρχικών συνθηκών*. Επομένως η συνάρτηση σφάλματος έχει δύο είδη όρων: τα σφάλματα της ΔΕ εξίσωσης L_{PDE} και τα σφάλματα στα εξωτερικά δεδομένα L_{DATA}. Η εκπαίδευση έγινε με εξωτερικά δεδομένα από τις τροχιές Lagrange με υψηλό θόρυβο (20%).

Επίσης θα πρέπει να υπενθυμίσουμε ότι χρησιμοποιήθηκε το σύστημα 2ης τάξης και επομένως τα υπόλοιπα της Δ.Ε είχαν την μορφή $\ddot{x}_i - G m_j \frac{(x_j - x_i)}{(r_{12}+\varepsilon)^3}$ όπου ε μικρή σταθερά στον παρονομαστή ώστε να αποφύγουμε διαίρεση με τον μηδέν.

Αρχικά εκπαιδεύσαμε ένα δίκτυο με 3 κρυφά στρώματα και 64 νευρώνες ανά στρώμα για 120000 εποχές με optimizer adamW με ρυθμό μάθησης 0.001 και weight_decay=0.004. Θέσαμε $\varepsilon = 10^{-5}$ για αν αποφύγουμε τα πολύ υψηλά υπόλοιπα κατά την αρχή στης εκπαίδευσης. Ο αλγόριθμος adamW είναι μια παραλλαγή του adam που χρησιμοποιεί απόσβεση συντελεστή βαρών weight_decay ανεξάρτητο της κλίσης. Επειδή στην συγκεκριμένη περίπτωση οι κλίσεις των υπόλοιπων της Δ.Ε επικρατούν επιλέξαμε τον συγκεκριμένο αλγόριθμο. Παρακάτω δίνουμε τα αποτελέσματα της εκπαίδευσης.



Σχήμα 80: Τροχιές - Pinn με εξωτερικά δεδομένα χωρίς ICs - adamW



Σχήμα 81: Απώλειες - Pinn με εξωτερικά δεδομένα χωρίς ICs - adamW

Στο σχήμα 80 παρατηρούμε ένα ενδιαφέρον φαινόμενο. Αν και η εκπαίδευση φαίνεται να απέτυχε το pinn έχει τοποθετήσει αρχικά τα σώματα σε μία ευθεία (μαύρες κουκίδες στο διάγραμμα). Αυτό ελέγχθηκε μετρώντας τις αποστάσεις των σωμάτων d_{12}, d_{13}, d_{23} και επιβεβαιώθηκε ότι $d_{13}+d_{23}\approx d_{12}$ με ακρίβεια δύο δεκαδικών ψηφίων. Επίσης το σώμα 3 είναι ακίνητο στο κέντρο και τα άλλα δύο περιφέρονται γύρω από αυτό. Αυτή όμως είναι η διαμόρφωση των περιοδικών τροχιών Euler που μελετήσαμε στις παραγράφους 4.2.3 και 4.2.4. Επομένως η έλλειψη αρχικών συνθηκών οδήγησε το pinn να ανακαλύψει μια νέα απλούστερη περιοδική τροχιά. Υπάρχουν αρκετές μελέτες σχετικά με την δυνατότητα των pinns να δίνουν πολλαπλές λύσεις σε προβλήματα ανάλογα με την αρχική διαμόρφωση του δικτύου καθώς και άλλων παραμέτρων. Ενδεικτικά αναφέρουμε μία πρόσφατη δημοσίευση (Zou κ.ά., 2025).

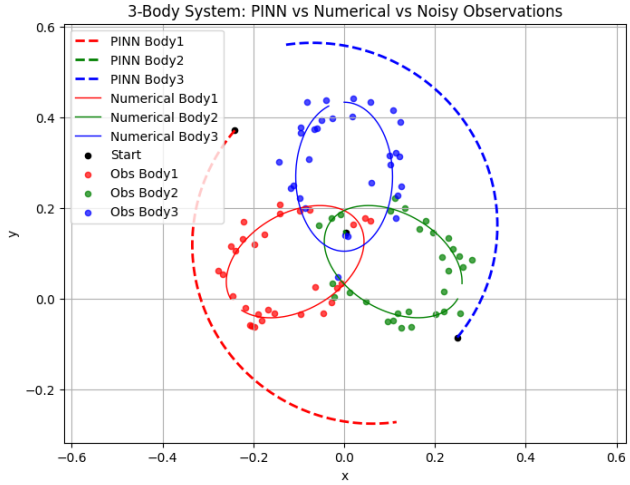
Στο σχήμα 81 που δίνει τις συνολικές απώλειες παρατηρούμε ότι αρχικά οι απώλειες είναι πολύ υψηλές (περίπου 10^9). Αυτό οφείλεται στις αρχικές τιμές των βαρών και πολώσεων του δικτύου. Δεδομένου ότι η έξοδος ενός νευρώνα είναι $\sigma(\sum_{i=1}^{n} w_i x_i + b)$ και αφού η συνάρτηση ενεργοποίησης που χρησιμοποιούμε δίνει $\sigma(0)=\tanh(0)=0$ τότε αν οι αρχικές πολώσεις είναι μηδέν όλοι οι νευρώνες θα δώσουν μηδενικές εξόδους την χρονική στιγμή $t=0$. Επομένως οι πολύ υψηλές τιμές στις απώλειες οφείλονται στο ότι το δίκτυο αρχικοποιείται με μηδενικές πολώσεις (biases) με αποτέλεσμα την στιγμή $t=0$ όλες οι έξοδοι του δικτύου να δίνουν μηδέν.

Όμως οι έξοδοι του δικτύου χρησιμοποιούνται στον παρανομαστή των υπολοίπων της Δ.Ε $Res_i=\ddot{x}_i-G m_j\frac{(x_j-x_i)}{(r_{12}+\varepsilon)^3}$. Η μικρή παράμετρος ε αποτρέπει την διαίρεση με μηδέν όμως τα υπόλοιπα παραμένουν πολύ υψηλά αφού $\varepsilon \ll 1$. Γενικά τα πολύ υψηλά υπόλοιπα της Δ.Ε δημιουργούν πρόβλημα κατά την εκπαίδευση διότι τα υπόλοιπα των εξωτερικών δεδομένων είναι συνήθως μικρά και επομένως το δίκτυο αγνοεί τα εξωτερικά δεδομένα τουλάχιστον κατά τις πρώτες εποχές εκπαίδευσης.

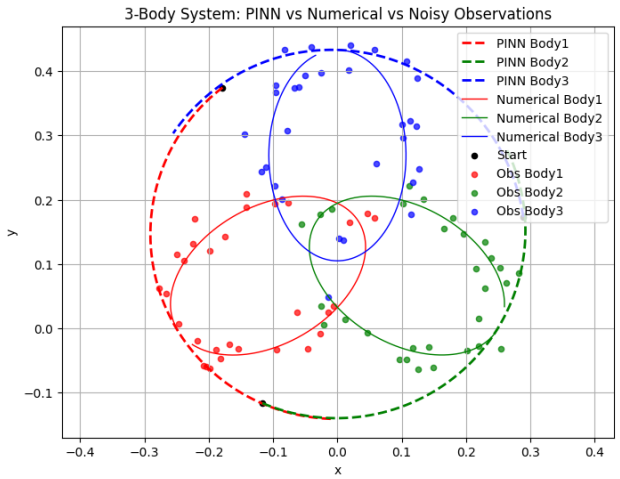
Ένας τρόπος να αντιμετωπισθεί το πρόβλημα που προκαλεί η έλλειψη αρχικών συνθηκών είναι να θέσουμε μη μηδενικές πολώσεις στο στρώμα εξόδου του δικτύου ώστε να μας δώσει διαφορετικές τιμές εξόδων την στιγμή $t=0$. Επειδή οι αρχικές τιμές των πολώσεων μπορεί να επηρεάσουν την εκπαίδευση του δικτύου επιλέγουμε μικρές τιμές τάξης 10^{-1}. Ο κώδικας που αλλάζει τις τιμές των πολώσεων στο στρώμα εξόδου δίνεται στο *Παράρτημα Η.*

Θέσαμε τα bias στους έξι νευρώνες του στρώματος εξόδου ως εξής: [-0.2, 0.2, 0, 0, 0.2, -0.2] . Επαναλάβαμε το προηγούμενο test με ίδιες παραμέτρους αλλάζοντας μόνο την μικρή παράμετρο σε $\varepsilon = 10^{-9}$ αφού δεν έχουμε πλέον προβλήματα μηδενισμού του παρονομαστή την στιγμή $t = 0$. Για την αρχικοποίηση των βαρών χρησιμοποιήσαμε την κατανομή ***Glorot Uniform*** που είναι η πιο συνηθισμένη για την εκπαίδευση pinns. Μετά από 40000 εποχές εκπαίδευσης πήραμε ξανά μια περιοδική τροχιά Euler (Σχήμα 82). Αλλάξαμε την κατανομή σε ***Glorot Normal*** και επαναλάβαμε το πείραμα και αυτήν την φορά το pinn έδωσε μια περιοδική τροχιά Lagrange (Σχήμα 83). Το pinn τοποθέτησε τα τρία σώματα αρχικά στις κορυφές ισόπλευρου τριγώνου. Παρακάτω φαίνονται οι σχετικές αποστάσεις των σωμάτων που επιβεβαιώνουν ότι σχηματίζουν ισόπλευρο τρίγωνο με ακρίβεια ενός δεκαδικού ψηφίου.

length-12	0.49468699438039304
length-13	0.5091119020678815
length-23	0.49849160451304597



Σχήμα 82: Τροχιές - Pinn με εξωτερικά δεδομένα χωρίς ICs -Glorot Uniform

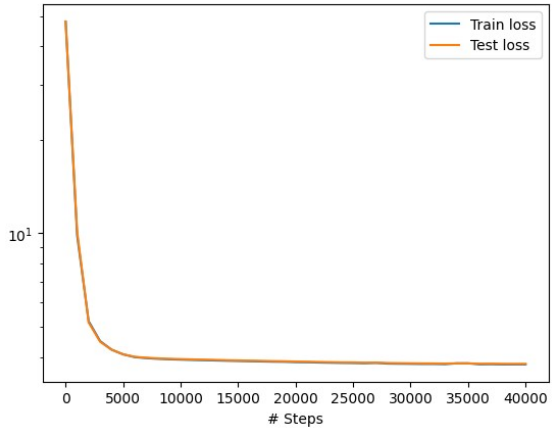


Σχήμα 83: Τροχιές - Pinn με εξωτερικά δεδομένα χωρίς ICs - Glorot Normal

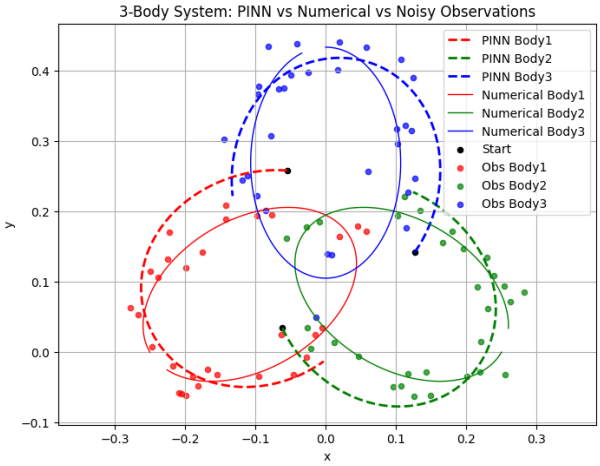
Επομένως καταλήγουμε στο συμπέρασμα ότι τα pinns μπορούν να δώσουν νέες περιοδικές λύσεις που εξαρτώνται από την αρχική κατανομή των βαρών του δικτύου. Αυτές οι παρατηρήσεις ανοίγουν ένα νέο πεδίο έρευνας στα pinns για την εύρεση περιοδικών λύσεων στο πρόβλημα των τριών σωμάτων.

Αν και κάναμε ενδιαφέρουσες παρατηρήσεις δεν έχουμε καταφέρει να βρούμε μια λύση που να ταιριάζει ικανοποιητικά με τα εξωτερικά δεδομένα. Το πρόβλημα φαίνεται να είναι οι γενικά υψηλότερες κλίσεις των υπολοίπων της διαφορικής εξίσωσης σε σχέση με τα υπόλοιπα των

εξωτερικών δεδομένων. Για να εξισορροπήσουμε αυτήν την διαφορά θα χρησιμοποιήσουμε adam optimizer με κατάλληλα βάρη σε κάθε όρο. Θέσαμε βάρη 0.1 στους όρους L_{PDE} και 40 στους όρους L_{DATA}. Για αρχικοποίηση χρησιμοποιήθηκε η κατανομή Glorot Uniform και ο ρυθμός μάθησης ήταν 10^{-4}. Μετά από 40000 εποχές έχουμε τα παρακάτω αποτελέσματα.



Σχήμα 85: Απώλειες - Pinn με εξωτερικά δεδομένα χωρίς ICs -Glorot Uniform-adam



Σχήμα 84: Τροχιές-Pinn με εξωτερικά δεδομένα χωρίς ICs -Glorot Uniform-adam

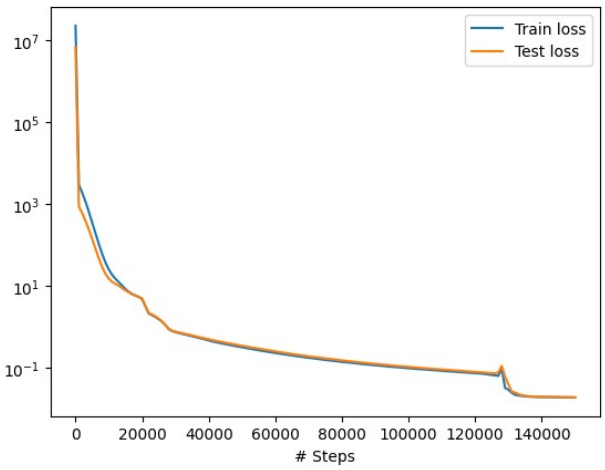
Από το διάγραμμα απωλειών παρατηρούμε ότι μετά από 6000 εποχές εκπαίδευσης δεν έχουμε περαιτέρω βελτίωση. Από το διάγραμμα τροχιών φαίνεται ότι έχουμε μια καλή προσέγγιση δεδομένου του μεγάλου θορύβου και της έλλειψης αρχικών συνθηκών. Επίσης παρατηρούμε την εντυπωσιακή ικανότητα του pinn να τοποθετεί αρχικά τα σώματα στις κορυφές ισόπλευρου τριγώνου χωρίς καμία πληροφορία για τις αρχικές συνθήκες του προβλήματος. Παρόλα αυτά μετά από αρκετά πειράματα φαίνεται ότι το αποτέλεσμα δεν είναι εύκολο να βελτιωθεί παρά τις διάφορες δοκιμές στα βάρη , τον ρυθμό μάθησης κ.τ.λ.

Για να πετύχουμε καλύτερα αποτελέσματα πιθανόν χρειάζεται μια λιγότερο στατική μέθοδος για την διαμόρφωση των βαρών και την εξομάλυνση των υψηλών κλίσεων στους όρους L_{PDE}. Για τον καλύτερο έλεγχο των κλίσεων εισάγουμε μία παράμετρο C με δυνατότητα να εκπαιδεύεται (trainable variable) και πολλαπλασιάζουμε τα υπόλοιπα της Δ.Ε με τον όρο $1/C^2$. Επομένως τα υπόλοιπα γίνονται $\left(\ddot{x}_i - G m_j \frac{(x_j - x_i)}{(r_{12}+\varepsilon)^3}\right)\frac{1}{C^2} = \frac{1}{C^2} Res_{PDE}$ και οι όροι σφαλμάτων (L2 loss terms) $\frac{1}{C^4}\frac{1}{n}\sum_{i=1}^{n}\left(\ddot{x}_i - G m_j \frac{(x_j - x_i)}{(r_{12}+\varepsilon)^3}\right)^2 = \frac{1}{C^4} L_{PDE}$.

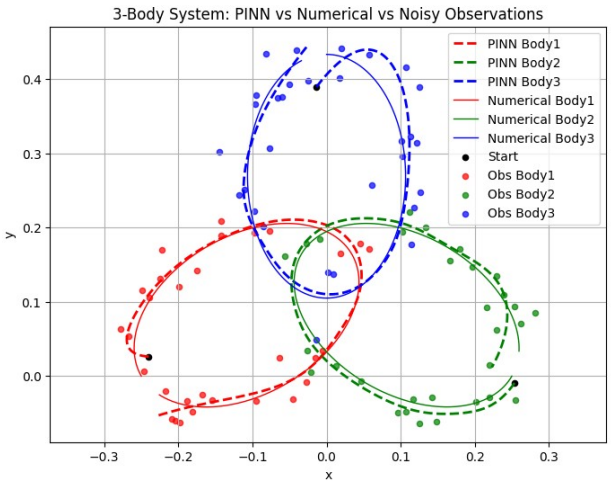
Αφού η παράμετρος C εκπαιδεύεται το pinn θα υπολογίζει και τις κλίσεις δηλ. $\nabla_C \frac{1}{C^4} L_{PDE} = -4 \frac{1}{C^3} L_{PDE}$. Επομένως αν θέσουμε μια αρχικά θετική τιμή ώστε $C>1$ τότε η παράμετρος θα αυξάνεται διαρκώς με αποτέλεσμα όσο αυξάνονται οι εποχές εκπαίδευσης το δίκτυο να εκπαιδεύεται περισσότερο στα δεδομένα διότι η αύξηση της C θα μειώνει τεχνητά τα σφάλματα των όρων της Δ.Ε.

Εδώ χρησιμοποιούμε μια συνάρτηση της παραμέτρου C και πολλαπλασιάζουμε τα υπόλοιπα της Δ.Ε $\left(\ddot{x}_i - G m_j \frac{(x_j - x_i)}{(r_{12}+\varepsilon)^3}\right) f(C) = f(C) Res_{PDE}$. Επιλέξαμε $f(C) = \frac{1}{C^2}$ που μάλλον δεν είναι η βέλτιστη συνάρτηση όμως παρόλα αυτά η επιλογή μας δίνει σημαντική ευελιξία σε σχέση με τα στατικά βάρη.

Για το πείραμα με την παράμετρο C χρησιμοποιήθηκε ίδιο δίκτυο ,optimizer και ρυθμός μάθησης με τα προηγούμενα παραδείγματα. Αλλάξαμε τα στατικά βάρη σε 0.1 για τους όρους PDE και 10 για τους όρους δεδομένων. Επίσης δεν αλλάξαμε τα bias του δικτύου. Η αρχική τιμή της παραμέτρου ήταν $C=3$. Μετά από 150000 εποχές εκπαίδευσης πήραμε τα παρακάτω αποτελέσματα.



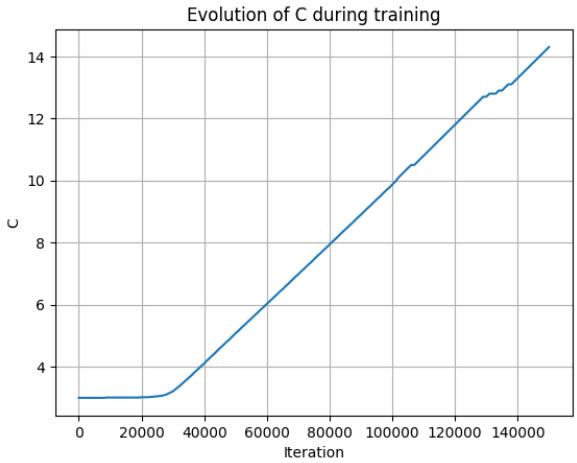
Σχήμα 87: Απώλειες - Pinn με εξωτερικά δεδομένα χωρίς ICs - Trainable parameter C



Σχήμα 86: Τροχιές- Pinn με εξωτερικά δεδομένα χωρίς ICs - Trainable parameter C

Από το σχήμα 86 που δίνει την πρόβλεψη του pinn με διακεκομμένες γραμμές φαίνεται ότι έχουμε πολύ καλύτερα αποτελέσματα και μάλιστα συγκρίσιμα με το αποτέλεσμα του σχήματος 78 όπου είχαμε επιβάλει τις αρχικές συνθήκες με hard constrains. Τα σημεία με τα μεγαλύτερα σφάλματα παρατηρούνται στην αρχή και στο τέλος των τροχιών. Η καλύτερη πρόβλεψη είναι του σώματος 2 (πράσινες διακεκομμένες γραμμές) που πρόβλεψε σωστά τις αρχικές συνθήκες του προβλήματος.

Επομένως η παραμόρφωση στις τροχιές φαίνεται να οφείλεται στην επιλογή της αρχικής συνθήκης την στιγμή $t=0$ αλλά και στο σφάλμα λόγω θορύβου στο τέλος των τροχιών. Επίσης από το διάγραμμα (σχήμα 88) παρατηρούμε ότι η παράμετρος C μετά από τις πρώτες 25000 εποχές αυξάνεται σταθερά με αργό ρυθμό περίπου 2 /20000 εποχές δηλ. 0.1 ανά 1000 εποχές εκπαίδευσης. Αυτή η συμπεριφορά είναι επιθυμητή αφού το δίκτυο εκπαιδεύεται αργά άλλα σταθερά προς τα εξωτερικά δεδομένα.

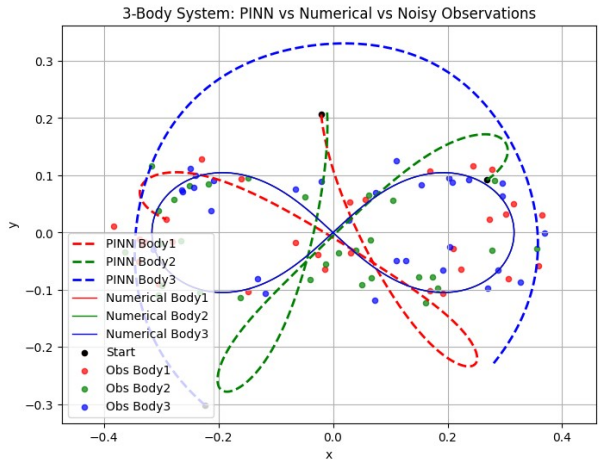


Σχήμα 88: Μεταβολή παραμέτρου C κατά την εκπαίδευση

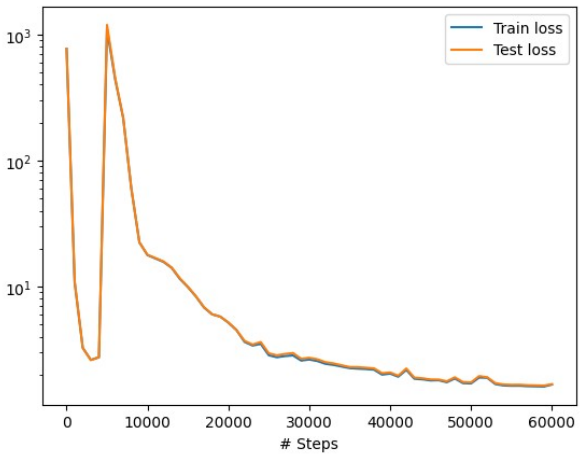
4.3.3 Αναζήτηση περιοδικών λύσεων στο πρόβλημα των τριών σωμάτων

Στην προηγούμενη παράγραφο παρατηρήσαμε ότι το pinn με εξωτερικά δεδομένα με θόρυβο *και χωρίς αρχικές συνθήκες* φαίνεται να βρίσκει νέες αρχικές συνθήκες για περιοδικές τροχιές. Στην παρούσα παράγραφο θα εξερευνήσουμε αυτήν την δυνατότητα των pinns.

Αρχικά εκπαιδεύσαμε ένα pinn με εξωτερικά δεδομένα από την τροχιά figure 8 με θόρυβο 20%. Χρησιμοποιήσαμε το δίκτυο 3x64 που χρησιμοποιήσαμε συχνά στην εργασία , συνάρτηση ενεργοποίησης sin , initializer Glorot Normal, adamW optimizer με ρυθμό μάθησης 0.001 και weight_decay=0.004. Επίσης θέσαμε βάρη 1 για τους όρους L_{PDE} και 10 για τα L_{DATA}. Μετά από 60000 εποχές εκπαίδευσης πήραμε τα παρακάτω αποτελέσματα:



Σχήμα 89: Έξοδος Pinn- δεδομένα με θόρυβο -τροχιά figure 8



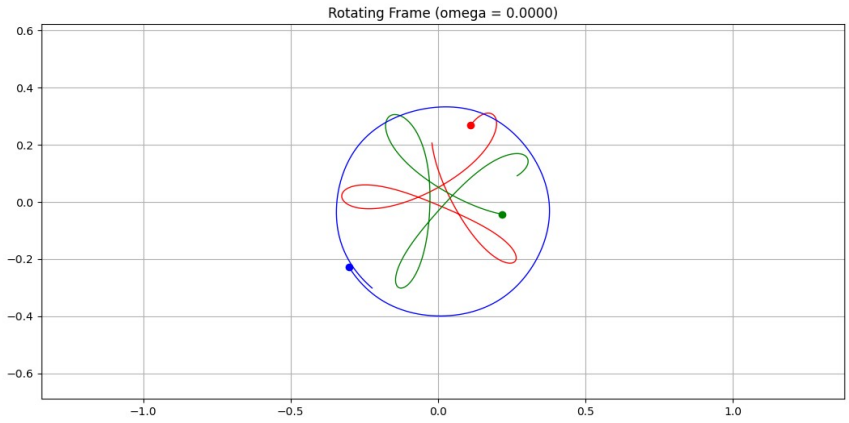
Σχήμα 90: Απώλειες δεδομένα με θόρυβο -τροχιά figure 8

Από την έξοδο του Pinn (σχήμα 89) παρατηρούμε ότι το pinn αντί να προσεγγίσει την τροχιά των δεδομένων (figure-8) βρήκε μία άλλη τροχιά που μοιάζει με την οικογένεια περιοδικών τροχιών τροχιών BHH. Οι αρχικές συνθήκες που έδωσε το δίκτυο είναι οι παρακάτω:

pinn inital conditions...

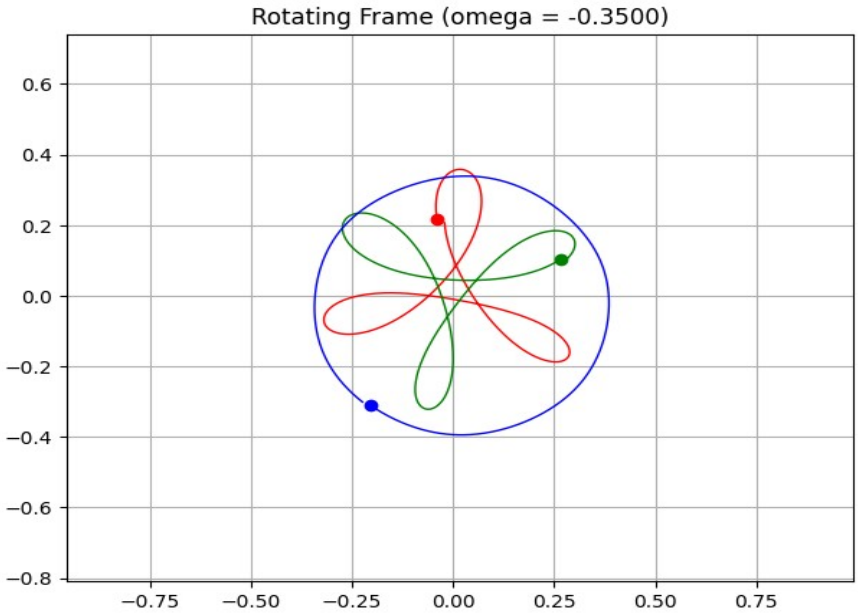
	x0,	y0,	vx0 ,	vy0
body_1	-0.02060754579306967	0.2062820548044174	0.1712777191418524	-1.6498101110265813
body_2	0.26842952279231386	0.09176782702268939	0.9394626194134084	0.6423382272799922
Body_3	-0.22364327446127136	-0.301289552725747	-1.1534225119266874	1.0107518821541785

Πήραμε τις αρχικές θέσεις και ταχύτητες που έδωσε το δίκτυο και τις χρησιμοποιήσαμε σαν αρχικές συνθήκες στην αριθμητική λύση και πήραμε το παρακάτω αποτέλεσμα:



Σχήμα 91: Τροχιά με βάση τις αρχικές συνθήκες του Pinn

Επειδή η περιοδικότητα της βασικής τροχιάς ΒΗΗ φαίνεται σε περιστρεφόμενο σύστημα αναφοράς (Liao κ.ά., 2022) χρησιμοποιήσαμε την αριθμητική επίλυση σε στρεφόμενο σύστημα αναφοράς με γωνιακή συχνότητα $\omega = -0.35$ και η τροχιά που πήραμε μοιάζει να είναι πολύ κοντά στην κλασσική τροχιά ΒΗΗ όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 92: Πρόβλεψη pinn σε στρεφόμενο σύστημα αναφοράς

Για να βεβαιωθούμε ότι είναι πράγματι κοντά στην περιοδική τροχιά χρησιμοποιήσαμε τις αρχικές συνθήκες του pinn και με μία μέθοδο ελαχίστων τετραγώνων που προσφέρει η βιβλιοθήκη scipy (**scipy.optimize.least_squares**) και αναζητήσαμε αρχικές συνθήκες που δίνουν περιοδικότητα με μεγάλη ακρίβεια. (Μία παρόμοια μέθοδο χρησιμοποιείται στην δημοσίευση των Liao κ.ά 2022 που αναφέραμε παραπάνω)

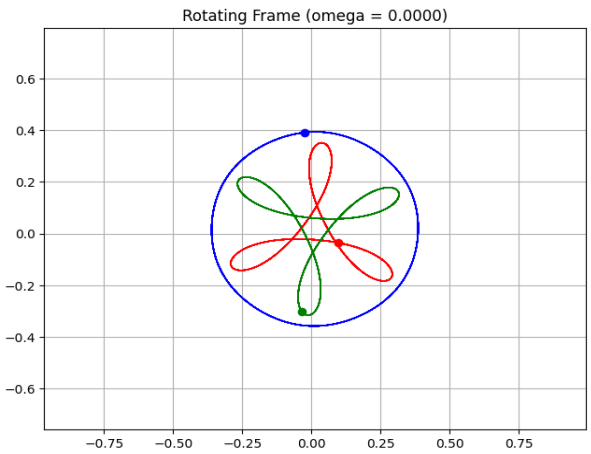
Πήραμε τις παρακάτω αρχικές συνθήκες:

```
Function evaluations 24, initial cost 1.3246e+00, final cost 4.3908e-19, first-
order optimality 4.73e-09.
Refined period: 1.4945932398075663
Error norm: 9.371070933416611e-10

Refined initial conditions:

  x1 = -0.0070013764466178
  y1 = 0.2173855957217074
  x2 = 0.2767254658098035
  y2 = 0.1005626572159539
  x3 = -0.2332709986696099
  y3 = -0.2638553655152565
 vx1 = 0.1896126238566085
 vy1 = -1.6360945335855852
 vx2 = 0.9989333517577658
 vy2 = 0.6272685875995790
 vx3 = -1.1885459756146692
 vy3 = 1.0088259459859923
```

Στο παρακάτω σχήμα φαίνεται η τροχιά που δίνουν οι νέες αρχικές συνθήκες που πήραμε με την μέθοδο προσέγγισης περιοδικής λύσης με ελάχιστα τετράγωνα.

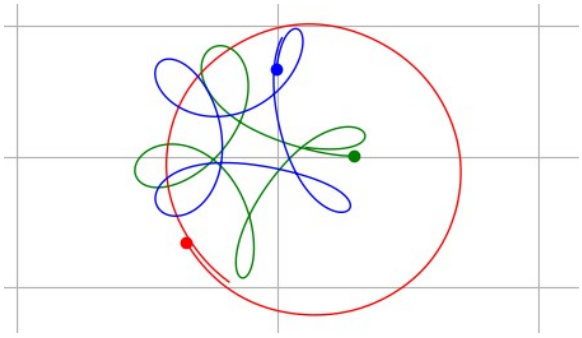


Σχήμα 93: Τροχιά με τις αρχικές συνθήκες που πήραμε με μέθοδο ελαχίστων τετραγώνων

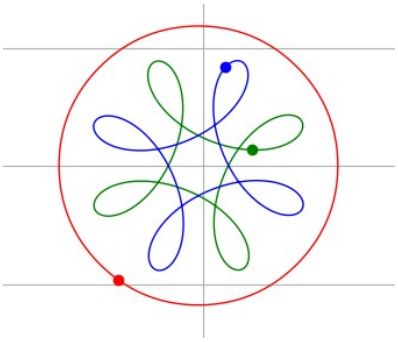
Στο σχήμα 93 φαίνεται το αποτέλεσμα με τις αρχικές συνθήκες που προέκυψαν από την μέθοδο ελαχίστων τετραγώνων ξεκινώντας από τις αρχικές συνθήκες του pinn και υποθέτοντας μια αρχική περίοδο 1.5. Η τροχιά είναι ξεκάθαρα η βασική τροχιά των Broucke (1975), Hadjidemetriou (1975) and Hénon (1976) (BHH). Εδώ θα πρέπει να τονίσουμε ότι για την αριθμητική μέθοδο που χρησιμοποιήσαμε είναι σημαντικό να δώσουμε μια αρχική πρόβλεψη για την περίοδο κοντά στην πραγματική αλλιώς μπορεί να πάρουμε μη φυσικά αποδεκτά αποτελέσματα π.χ αρνητική περίοδο ή σχεδόν μηδενική περίοδο. Ο κώδικάς που χρησιμοποιήθηκε για την εύρεση των περιοδικών αρχικών συνθηκών δίνεται στο *Παράρτημα Θ*.

Το αποτέλεσμα είναι σημαντικό διότι ξεκινώντας από μία τροχιά figure 8 πήραμε μια τροχιά BHH που ανήκει σε διαφορετική οικογένεια τροχιών. Αυτό αποτελεί ένδειξη ότι τα pinns μπορούν να ανακαλύψουν νέες τροχιές και όχι απλά παραλλαγές τροχιών που ανήκουν στην ίδια οικογένεια.

Στα επόμενα πειράματα χρησιμοποιήσαμε δεδομένα με θόρυβο από την κλασσική τροχιά BHH (Liao κ.ά., 2022) και αλλάζοντας την συνάρτηση ενεργοποίησης σε tanh και τον initializer σε Glorot Unoniform από την αρχική τροχιά BHH με τρεις λοβούς πήραμε με την ίδια μέθοδο μια περιοδική τροχιά με τέσσερις λοβούς όπως φαίνεται παρακάτω:

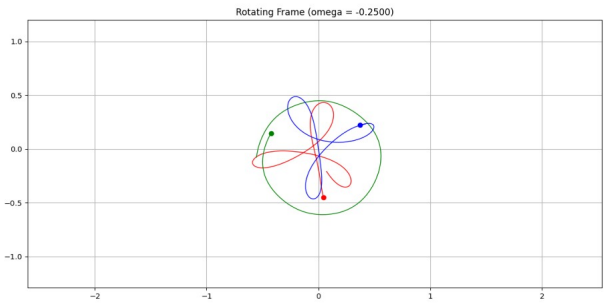


Σχήμα 95: Αρχική πρόβλεψη Pinn

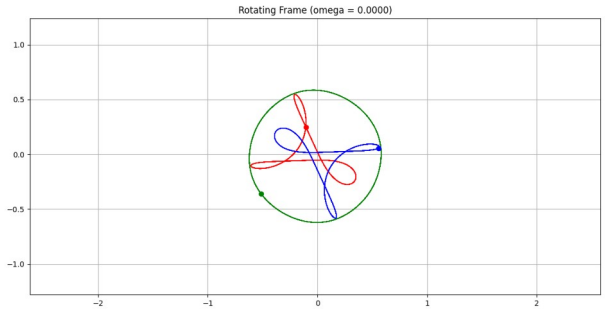


Σχήμα 94: Τροχιά από αρχικές συνθήκες με μέθοδο ελαχίστων τετραγώνων

Αλλάζοντας την αρχικοποίηση του δικτύου (δηλ. αλλάζοντας το seed που παράγει τα αρχικά βάρη) πήραμε τις παρακάτω τροχιές:



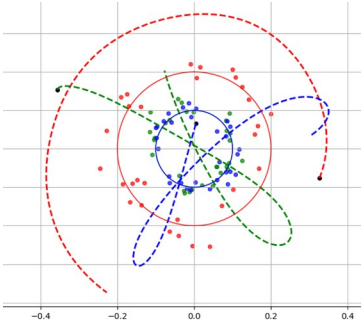
Σχήμα 97: Αρχική πρόβλεψη Pinn με νέα αρχικοποίηση



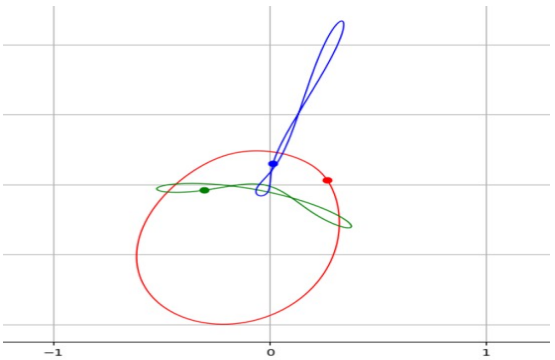
Σχήμα 96: Τροχιά από αρχικές συνθήκες με μέθοδο ελαχίστων τετραγώνων

Από τα παραπάνω αποτελέσματα μπορούμε να συμπεράνουμε ότι τα pinns μπορούν να μας δώσουν αρχικές συνθήκες που είναι "κοντά" σε πραγματικές περιοδικές τροχιές. Παρατηρήσαμε ότι αυτή συμπεριφορά εξαρτάται τόσο από τα εξωτερικά δεδομένα όσο και από την αρχικοποίηση του δικτύου και την συνάρτηση ενεργοποίησης. Για αν διερευνήσουμε τον ρόλο των εξωτερικών δεδομένων κάναμε μια σειρά από πειράματα όπου τα *εξωτερικά δεδομένα δεν προέρχονταν από πραγματικές λύσεις* αλλά είχαν κάποιες συμμετρίες π.χ ομόκεντροι κύκλοι κ.τ.λ.

Χρησιμοποιώντας ίδια διαμόρφωση με το προηγούμενο πείραμα αλλάξαμε τα εξωτερικά δεδομένα με θόρυβο σε τρείς ομόκεντρες κυκλικές τροχιές με ακτίνες 0.2 , 0.1 και 0.1 δηλ. Οι δύο τελευταίες τροχιές είχαν ίδια ακτίνα. Η πρώτη τροχιά είχε δεξιόστροφη κίνηση και οι άλλες δύο αριστερόστροφη ώστε να έχουμε μηδενική ορμή και στροφορμή. Παρ όλες τις συμμετρίες οι τροχιές δεν αντιστοιχούν σε κάποια φυσική λύση του προβλήματος τριών σωμάτων. Πήραμε τα παρακάτω αποτελέσματα:



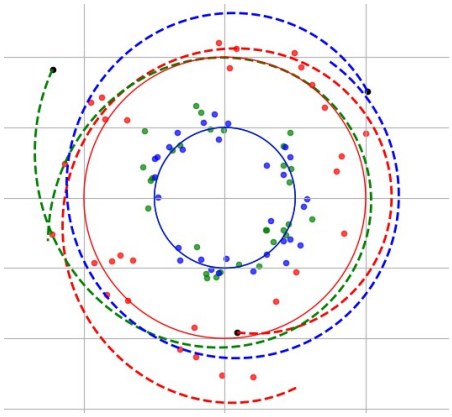
Σχήμα 98: Pinn με δεδομένα από μη πραγματική τροχιά



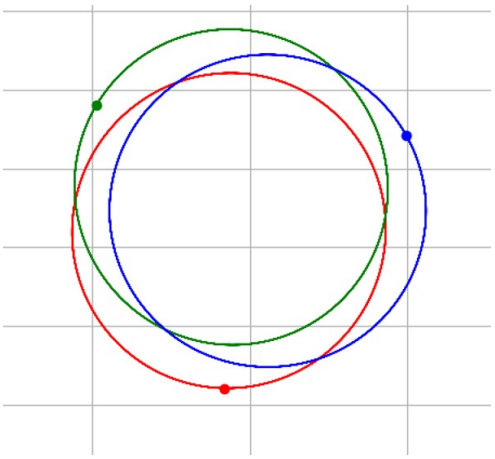
Σχήμα 99: Νέες αρχικές συνθήκες - ελάχιστα τετράγωνα

Αν και η τροχιά του σχήματος 99 που προέκυψε με ελάχιστα τετράγωνα δεν φαίνεται να μοιάζει με την αρχική πρόβλεψη του pinn είναι μία περιοδική τροχιά.

Αλλάζοντας την αρχική διαμόρφωση του δικτύου πήραμε τα παρακάτω αποτελέσματα:



Σχήμα 101: Πρόβλεψη Pinn

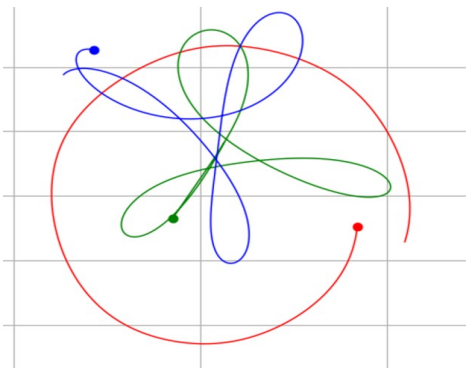


Σχήμα 100: Τροχιά με την μέθοδο ελαχίστων τετραγώνων

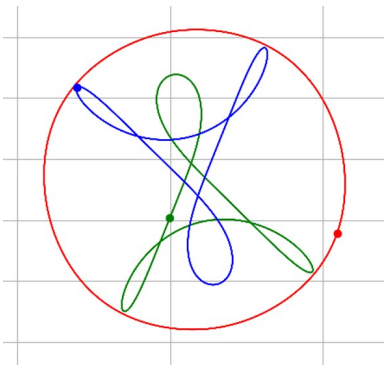
Εδώ παρατηρούμε ότι η πρόβλεψη του δικτύου και η περιοδική τροχιά που προέκυψε με την προσέγγιση ελαχίστων τετραγώνων μοιάζουν αρκετά.

Παρατηρήσαμε ότι η τροχιά του σχήματος 100 εμφανιζόταν στα περισσότερα πειράματα που κάναμε με αυτήν την διαμόρφωση.

Κατόπιν χρησιμοποιήσαμε δεδομένα με θόρυβο με ομόκεντρους κύκλους με διαφορετικές ακτίνες 0.6 , 0.2 , 0.1 όπου εκτός από την γεωμετρική συμμετρία δεν είχαμε κάποια άλλη φυσική συμμετρία (π.χ μηδενισμό ορμής ή στροφορμής) και πήραμε πάλι την ίδια περιοδική τροχιά που είχαμε πάρει ξεκινώντας από τις τροχιές ΒΗΗ.



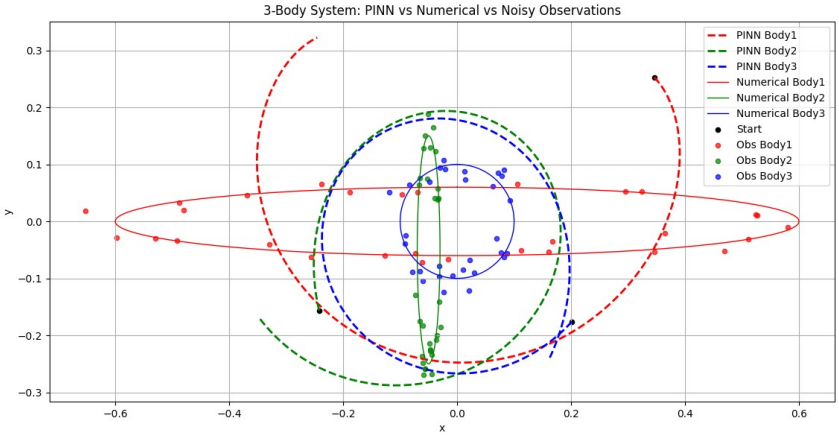
Σχήμα 103: Pinn - πρόβλεψη δεδομένα από ομόκεντρους κύκλους



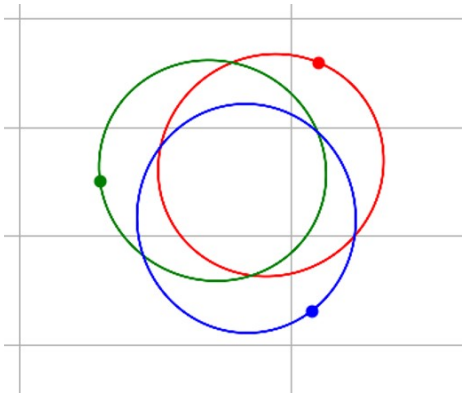
Σχήμα 102: Περιοδική λύση με μέθοδο ελαχίστων τετραγώνων

Από τα παραπάνω φαίνεται ότι τα εξωτερικά δεδομένα περιορίζουν την λύση σε μια περιοχή του χώρου και επιβάλουν κάποιες γεωμετρικές συμμετρίες. *Επομένως δεν είναι απαραίτητο να έχουμε πραγματικές λύσεις στην διάθεσή μας για να πάρουμε περιοδικές τροχιές*. Αυτό είναι ιδιαίτερα σημαντικό αν μπορεί να επεκταθεί σε χαοτικά συστήματα για τα οποία δεν έχουμε στην διάθεσή μας κάποια περιοδική λύση.

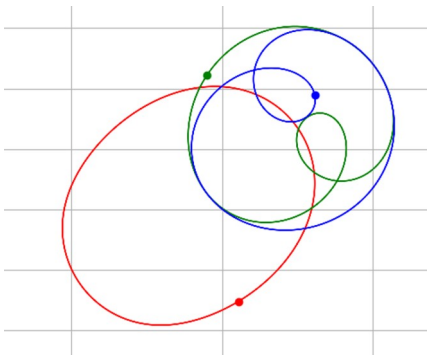
Τα τελευταία πειράματα έγιναν με μη φυσικές τροχιές που έχουν ελλειπτικά σχήματα. Τα πειράματα έγινα χωρίς αλλαγή στην διαμόρφωση του δικτύου με αλλαγή των seeds στις μηχανές τυχαίων αριθμών που επηρεάζουν την αρχικοποίηση του δικτύου. Παρακάτω έχουμε τα σχήματα από τα τελικά αποτελέσματα:



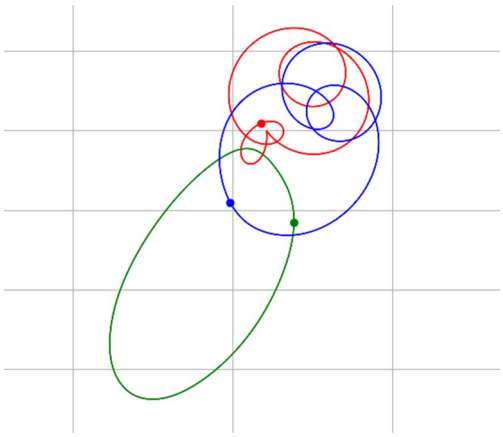
Σχήμα 104: Pinn με δεδομένα από μη πραγματικές ελλειπτικές τροχιές



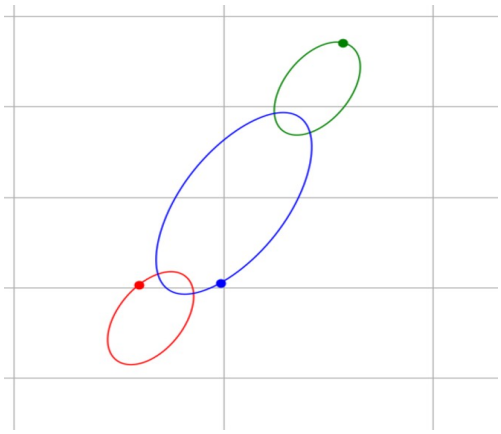
Σχήμα 105: Ελάχιστα τετράγωνα - Λύση-1



Σχήμα 106: Ελάχιστα τετράγωνα - Λύση-2



Σχήμα 107: Ελάχιστα τετράγωνα - Λύση-3



Σχήμα 108: Ελάχιστα τετράγωνα - Λύση-4

Οι λύσεις στα σχήματα 105 έως 108 προέκυψαν από λύσεις που έδωσαν τα pinns για αρχικές – μη φυσικές - ελλειπτικές τροχιές με χρήση της μεθόδου ελαχίστων τετραγώνων.

Θα πρέπει να αναφέρουμε ότι παρόμοια αποτελέσματα μπορούμε να πάρουμε και από αρχικά δεδομένα χωρίς θόρυβο όμως η προσθήκη κατάλληλου επιπέδου θορύβου φαίνεται να ευνοεί την πολλαπλότητα των τροχιών που προκύπτουν από μία αρχική διαμόρφωση. Επίσης έγιναν πειράματα με πολύ μεγάλο ποσοστό θορύβου >50% χωρίς σημαντικά αποτελέσματα. *Σε κάθε περίπτωση ο αριθμός των πειραμάτων δεν ήταν επαρκής για να αποφανθούμε αν κάποιο ποσοστό θορύβου δίνει καλύτερα αποτελέσματα.*

Από τα παραπάνω πειράματα μπορούμε να συμπεράνουμε ότι τα εξωτερικά δεδομένα περιορίζουν τις λύσεις που δίνει το pinn τόσο χωρικά όσο και γεωμετρικά. Όμως δεν είναι εύκολο να αιτιολογήσουμε γιατί τα pinns βρίσκουν λύσεις που είναι κοντά σε περιοδικές λύσεις του προβλήματος. Πιθανόν το χαοτικό πρόβλημα των τριών σωμάτων να έχει ιδιότητες που ευνοούν την εύρεση περιοδικών λύσεων. Για παράδειγμα πολλές οικογένειες περιοδικών λύσεων είναι συνεχείς μονοπαραμετρικές οικογένειες (Antoniadou κ.ά., 2010)

Αυτό που προκύπτει από τα πειράματα είναι ότι τα Pinns μπορούν να εντοπίσουν λύσεις που είναι κοντά σε πραγματικές περιοδικές λύσεις για το πρόβλημα των τριών σωμάτων χωρίς καμία ουσιαστική πληροφορία είτε με την μορφή αρχικών συνθηκών είτε με την μορφή δεδομένων μιας πραγματικής περιοδικής τροχιάς.

5 Συμπεράσματα

Πριν προχωρήσουμε στα συμπεράσματα θα πρέπει να αναφέρουμε ότι κατά την διάρκεια των πειραμάτων προτιμήσαμε τα πολλά πειράματα σε βάρος των πολύ καλών αποτελεσμάτων. Επιδιώξαμε καλά αποτελέσματα αλλά στις περισσότερες περιπτώσεις δεν έγινε χρήση των χρονοβόρων optimizers δεύτερης τάξης όπως ο L-BFGS για την τελειοποίηση των αποτελεσμάτων.

Τα περισσότερα πειράματα εκτελέστηκαν σε έναν Home computer με επεξεργαστή intel i7 και συνολική μνήμη RAM 8GB. Η μνήμη που δεσμεύτηκε για την εκτέλεση των περισσότερων πειραμάτων ήταν 0.5 GB που είναι οριακή. Η κρίσιμη παράμετρος για την γρήγορη εκτέλεση των πειραμάτων είναι η μνήμη και η δυνατότητα παράλληλης επεξεργασίας που προσφέρουν οι σύγχρονες κάρτες γραφικών (GPU). Δυνατότητα πρόσβασης σε εξελιγμένο hardware δίνεται δωρεάν από το google colab όμως δεν έγινε εκτεταμένη χρήση διότι οι GPUs είναι κατάλληλες για συστήματα με παραγώγους 1ης τάξης ενώ μπορεί να είναι πολύ αργές στους υπολογισμούς παραγώγων 2ης τάξης (Hessian). Για την εκτέλεση κάποιων παραδειγμάτων με συστήματα 2ης τάξης χρησιμοποιήθηκαν οι TPUs (tensor processing units) που προσφέρονται επίσης από το google colab και παρατηρήσαμε ότι ήταν 2 με 3 φορές γρηγορότερες από τον home computer.

Η βιβλιοθήκη deepxde ήταν πολύ χρήσιμη τόσο λόγω της εύκολης συγγραφής του κώδικα άλλα και λόγω των πολλών παραδειγμάτων όπου γίνεται χρήση προχωρημένων τεχνικών. Σε κάποιες περιπτώσεις χρησιμοποιήθηκε η βιβλιοθήκη tensorflow όπως για παράδειγμα για την χρήση του optimizer adamW που δεν έχει υλοποιηθεί μέχρι στιγμής στην βιβλιοθήκη deepxde. Θα πρέπει να σημειωθεί ότι χρησιμοποιήθηκε η μέγιστη ακρίβεια που παρέχει η βιβλιοθήκη tensorflow (64 bit) λόγω των πολύ μεγάλων τιμών στα υπόλοιπα των Δ.Ε ειδικά κατά τα πρώτα στάδια της εκπαίδευσης.

Επίσης όπως αναφέρθηκε και στο κείμενο δεν χρησιμοποιήθηκαν οι συμμετρίες των προβλημάτων που μειώνουν το πλήθος των Δ.Ε διότι σκοπός ήταν να αναγνωρισθούν και να αντιμετωπισθούν οι δυσκολίες στην εκπαίδευση των pinns με συστήματα πολλών Δ.Ε έτσι ώστε τα αποτελέσματα να είναι χρήσιμα και σε προβλήματα με μεγαλύτερο αριθμό σωμάτων όπως το σύστημα των τεσσάρων σωμάτων.

Το μεγαλύτερο πρόβλημα που αντιμετωπίσαμε στα συστήματα των δύο και τριών σωμάτων ήταν η δυσκολία των pinns στην μάθηση των αρχικών συνθηκών. Επειδή τα υπόλοιπα των όρων της Δ.Ε εξίσωσης είχαν μεγαλύτερες τιμές και κλίσεις σε σχέση με τις κλίσεις των όρων με τις αρχικές

συνθήκες η εκπαίδευση του δικτύου στις σωστές αρχικές συνθήκες αποδείχθηκε ιδιαίτερα δύσκολη. Ο νόμος του αντιστρόφου τετραγώνου της βαρυτικής δύναμης ($1/r^2$) έκανε ακόμα πιο δύσκολο το πρόβλημα λόγω των μεγάλων επιταχύνσεων όταν τα σώματα πλησίαζαν. Το πρόβλημα των αρχικών συνθηκών αντιμετωπίσθηκε με έναν μετασχηματισμό στην έξοδο του δικτύου (hard constrains) . Το επόμενο πρόβλημα ήταν οι πολλοί όροι στην συνάρτηση σφάλματος που δεν ήταν της ίδιας τάξης μεγέθους. Αυτό δυσκολεύει σημαντικά την εκπαίδευση διότι το δίκτυο δεν μαθαίνει ομαλά τους όρους ταχυτήτων και επιταχύνσεων των σωμάτων. Στο πρόβλημα των δύο σωμάτων με κατάλληλα βάρη πήραμε καλά αποτελέσματα όμως στα τρία σώματα οι όροι ήταν πολλαπλάσιοι και προτιμήσαμε να χρησιμοποιήσουμε το σύστημα 2ης τάξης.

Εδώ θα πρέπει να παρατηρήσουμε την θεμελιώδη διαφορά της επίλυσης Δ.Ε με αριθμητικές μεθόδους και με νευρωνικά δίκτυα όπως τα pinns. Ενώ κατά την αριθμητική επίλυση προτιμούμε ένα σύστημα 2ης τάξης να το μετατρέπουμε σε 1ης τάξης διότι έχουμε μεγαλύτερη ακρίβεια και καλύτερη σύγκλιση στα pinns αυτή η μέθοδος μπορεί να δημιουργήσει σοβαρά προβλήματα. Στην περίπτωση των τριών σωμάτων το σύστημα 1ης τάξης μας έδωσε 6 όρους επιταχύνσεων και 6 όρους ταχυτήτων. Επιπλέον οι όροι των επιταχύνσεων είχαν πολύ μεγαλύτερες τιμές λόγω του νόμου του αντιστρόφου τετραγώνου. Επομένως το pinn έπρεπε να εκπαιδευτεί σε 12 όρους με μεγάλες διαφορές μεταξύ τους. Αυτό έκανε την εκπαίδευση ιδιαίτερα δύσκολη. Για αυτόν τον λόγο ειδικά στο πρόβλημα των τριών σωμάτων προτιμήσαμε τα συστήματα 2ης τάξης που είχαν 6 όρους για τις Δ.Ε που ήταν όλοι όροι επιταχύνσεων και επομένως πιο ισορροπημένοι. Επίσης τα pinns έχουν συχνά δυσκολίες στην μάθηση των αρχικών συνθηκών ενώ οι αριθμητικές μέθοδοι δεν έχουν σοβαρό πρόβλημα τουλάχιστον όταν ο χρόνος ολοκλήρωσης είναι της τάξης της μίας περιόδου. Γι αυτό όπως αναφέραμε στις περισσότερες περιπτώσεις επιβάλλαμε τις αρχικές συνθήκες με έναν μετασχηματισμό στην έξοδο του δικτύου. Η πιο βασική διαφορά είναι ότι τα pinns μέσω της διαδικασίας της αυτόματης παραγώγισης μπορούν να υπολογίσουν παραγώγους σε οποιοδήποτε σημείο της τροχιάς ανεξάρτητα από το αν υπάρχουν άλλα σημεία κοντά του. Αντίθετα στις αριθμητικές μεθόδους οι παράγωγοι σε ένα σημείο υπολογίζονται με βάση τις τιμές των γειτονικών του σημείων μέσω ενός πλέγματος. Στα pinns τα σημεία εκπαίδευσης επιλέγονται τυχαία μέσω μιας συνάρτησης κατανομής. Οι χρονικές παράγωγοι ενός σημείου στα pinns υπολογίζονται ως παράγωγοι των εξόδων του νευρωνικού δικτύου ως προς τις εισόδους του και εξαρτώνται από όλα τα βάρη του δικτύου. Ένα νευρωνικό δίκτυο μαθαίνει τις Δ.Ε ενός προβλήματος ταυτόχρονα σε όλα τα σημεία εκπαίδευσης μέσα από μια διαδικασία εύρεσης ενός ολικού ελαχίστου μιας συνάρτησης σφάλματος.

Στις περισσότερες περιπτώσεις χρησιμοποιήσαμε 64 σημεία για εκπαίδευση και 100 διαφορετικά σημεία για έλεγχο. Παρατηρήσαμε ότι σε κάποιες περιπτώσεις (π.χ τροχιές Lagrange) ενώ το δίκτυο μάθαινε πολύ καλά την λύση στα 64 σημεία εκπαίδευσης είχε πολύ μεγάλα σφάλματα στα σημεία ελέγχου. Αυτό είναι ένα συχνό πρόβλημα και δείχνει ότι το δίκτυο δεν γενίκευσε και απλά βρήκε μια καλή προσέγγιση για λίγα σημεία του πεδίου λύσης. Αντιμετωπίσαμε το πρόβλημα με την τεχνική Residual-Based Adaptive Refinement (RAR) όπου βρίσκουμε τα σημεία με τα μεγαλύτερα σφάλμα και τα προσθέτουμε στο σύνολο σημείων εκπαίδευσης. Επίσης χρησιμοποιήσαμε μια μικρή παραλλαγή της μεθόδου όπου αντί για τα σημεία με τα μεγαλύτερα σφάλματα επιλέξαμε σημεία από ένα εύρος σημείων με 60% έως 95% χειρότερες απώλειες.

Στις περιοδικές τροχιές σχήματος οκτώ (figure 8) παρατηρήσαμε ένα άλλο ενδιαφέρον φαινόμενο. Ενώ η αριθμητική λύση είναι σταθερή για πολλές περιόδους και επίσης τα σώματα δεν πλησιάζουν πολύ κοντά το pinn είχε μεγάλη δυσκολία με το συγκεκριμένο πρόβλημα. Στις τροχιές figure 8 τα τρία σώματα εκτελούν την ίδια τροχιά σχήματος οκτώ με διαφορά φάσης 120^{o}. Περιμέναμε ότι λόγω της υψηλής συμμετρίας και των ομαλών επιταχύνσεων το pinn θα μπορούσε εύκολα να λύσει το πρόβλημα. Παρ όλα αυτά η υψηλή συμμετρία δυσκόλεψε το δίκτυο. Το πρόβλημα λύθηκε με έναν μετασχηματισμό στην είσοδο του νευρωνικού δικτύου που βοήθησε στην εκπαίδευση του δικτύου σε υψηλότερες συχνότητες. Εδώ δεν είχαμε ένα πρόβλημα με ιδιαίτερα υψηλές συχνότητες που είναι γνωστό ότι τα pinns εμφανίζουν πρόβλημα. Στην συγκεκριμένη περίπτωση φαίνεται ότι το πρόβλημα ήταν διαφορετικές συχνότητες με περίπου ίσα βάρη.

Στο τελευταίο μέρος της εργασίας εκπαιδεύσαμε το δίκτυο με εξωτερικά δεδομένα με υψηλό θόρυβο. Η σύγκριση με το απλό νευρωνικό δίκτυο έδειξε ότι τα pinns έχουν σαφές πλεονέκτημα ειδικά στην περίπτωση που έχουμε λίγα δεδομένα με θόρυβο. Επίσης έγινε προσπάθεια εκπαίδευσης με εξωτερικά δεδομένα με θόρυβο χωρίς γνώση των αρχικών συνθηκών του προβλήματος. Εδώ παρατηρήσαμε μια πολύ σημαντική ιδιότητα των pinns. Λόγω της έλλειψης αρχικών συνθηκών το δίκτυο βρήκε διαφορετικές λύσεις από αυτήν που περιμέναμε. Ενώ τα δεδομένα είχαν παραχθεί από μία λύση ελλειπτικών τροχιών Lagrange το pinn αρχικά έδωσε μια λύση περιοδικών τροχιών Euler τοποθετώντας τα τρία σώματα σε μια ευθεία με το μεσαίο σώμα ακίνητο. Όταν αλλάξαμε την αρχική κατανομή των βαρών του δικτύου το pinn έδωσε μία διαφορετική απλούστερη λύση τροχιών Lagrange όπου τα τρία σώματα βρίσκονται στις κορυφές ισόπλευρου τριγώνου και εκτελούν την ίδια κυκλική τροχιά. Τελικά καταφέραμε να προσεγγίσουμε την σωστή λύση χρησιμοποιώντας μια παράμετρο που εκπαιδεύονταν κατά την διάρκεια του πειράματος (trainable variable) και λειτουργούσε σαν ένα μεταβλητό βάρος που σταδιακά μείωνε τα υπόλοιπα της Δ.Ε ώστε το δίκτυο να αρχίσει να εκπαιδεύεται στα εξωτερικά δεδομένα.

Από τα παραπάνω μπορούμε να συμπεράνουμε ότι τα νευρωνικά δίκτυα ενημερωμένα με εξισώσεις φυσικής (Pinns) έχουν σαφές πλεονέκτημα σε σχέση με τα απλά νευρωνικά δίκτυα και μπορούν να δώσουν πολύ πιο αξιόπιστα αποτελέσματα όταν έχουμε να επεξεργασθούμε αραιά δεδομένα με θόρυβο.

Η σύγκριση των pinns με τις αριθμητικές μεθόδους δεν είναι τόσο απλή. Όπως αναφέραμε οι δύο μέθοδοι έχουν θεμελιώδεις διαφορές. Σαφώς οι αριθμητικές μέθοδοι μπορούν να δώσουν σε μικρότερο χρόνο αξιόπιστα αποτελέσματα. Επίσης έχουν αναπτυχθεί μαθηματικά εργαλεία που εξασφαλίζουν την σύγκλιση και την εκτίμηση του σφάλματος. Επομένως δεν μπορούμε να ισχυρισθούμε ότι τα νευρωνικά δίκτυα μπορούν να αντικαταστήσουν τις αριθμητικές μεθόδους. Όμως τα νευρωνικά δίκτυα έχουν ορισμένες σημαντικές ιδιότητες . Μπορούν να αντιμετωπίσουν προβλήματα μη καλά ορισμένα ή συστήματα ιδιαίτερα περίπλοκα για τα οποία έχουμε ελλειπή γνώση. Χρησιμοποιώντας πειραματικά δεδομένα και την γνώση που έχουμε σε μορφή διαφορικών εξισώσεων μπορούν να γενικεύσουν και να μας δώσουν χρήσιμες προβλέψεις για συστήματα που δεν έχουμε ικανοποιητικά μαθηματικά μοντέλα. Επίσης όπως παρατηρήσαμε τα pinns μπορούν να δώσουν διαφορετικές περιοδικές λύσεις σε χαοτικά συστήματα όπως το σύστημα των τριών σωμάτων. Επομένως τα pinns είναι χρήσιμο εργαλείο για την εξερεύνηση χαοτικών συστημάτων.

Δεδομένου ότι τα νευρωνικά δίκτυα και τα pinns είναι νέες περιοχές έρευνας τα αποτελέσματα που έχουν δώσει είναι πολλά υποσχόμενα. Θέματα όπως η σύγκλιση και η βέλτιστη εκπαίδευση των νευρωνικών δικτύων είναι ανοικτά και πιθανόν στο μέλλον να οδηγήσουν στην ανακάλυψη νέων μαθηματικών μεθόδων και θεωριών που να προσφέρουν καλύτερη κατανόηση της πολυπλοκότητας των φυσικών φαινομένων.

Βιβλιογραφία

Antoniadou, K. I., Voyatzis, G., & Kotoulas, T. (2010, Δεκέμβριος 21). *On the bifurcation and continuation of periodic orbits in the three-body problem*. arXiv.Org. https://doi.org/10.1142/S0218127411029720

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2020, Μάιος 2). *A survey on modern trainable activation functions*. arXiv.Org. https://doi.org/10.1016/j.neunet.2021.01.026

Babni, A., Jamiai, I., & Rodrigues, J. A. (2025). Error Estimates and Generalized Trial Constructions for Solving ODEs Using Physics-Informed Neural Networks. *Mathematical and Computational Applications*, *30*(6), 127. https://doi.org/10.3390/mca30060127

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). *Automatic differentiation in machine learning: A survey* (arXiv:1502.05767). arXiv. https://doi.org/10.48550/arXiv.1502.05767

Boughammoura, A. (2023). A Two-Step Rule for Backpropagation. *International Journal of Informatics and Applied Mathematics*, *6*(1), 57–69. https://doi.org/10.53508/ijiam.1265832

Chenciner, A., & Montgomery, R. (2000). *A remarkable periodic solution of the three-body problem in the case of equal masses* (arXiv:math/0011268). arXiv. https://doi.org/10.48550/arXiv.math/0011268

Fucheng, D., Wanjie, W., Ao, G., Xiaoqi, W., & Fan, W. (2025). *Gradient Descent Algorithm Survey* (arXiv:2511.20725). arXiv. https://doi.org/10.48550/arXiv.2511.20725

Ganga, S., & Uddin, Z. (2024). *Exploring Physics-Informed Neural Networks: From Fundamentals to Applications in Complex Systems* (arXiv:2410.00422). arXiv. https://doi.org/10.48550/arXiv.2410.00422

GEO5017—Machine Learning for the Built Environment. (χ.χ.). Ανακτήθηκε 31 Δεκέμβριος 2025, από https://3d.bk.tudelft.nl/courses/geo5017/lectures.html

Goldstein, H., Poole, C. P., & Safko, J. L. (2002). *Classical mechanics* (3ο έκδ.). Addison Wesley.

Hammad, M. M. (2024). *Artificial Neural Network and Deep Learning: Fundamentals and Theory* (arXiv:2408.16002). arXiv. https://doi.org/10.48550/arXiv.2408.16002

Hao, B., Braga-Neto, U., Liu, C., Wang, L., & Zhong, M. (2024, Απρίλιος 24). *Stability in Training PINNs for Stiff PDEs: Why Initial Conditions Matter*. arXiv.Org. https://arxiv.org/abs/2404.16189v3

Hysa, A., Priyadarshini, M. S., & Mahmoud, M. M. (2025). Numerical Solution of the System of Non-Linear Differential Equations of the Three-Body Problem in a Special Case. *European Modern Studies Journal*, *9*(1), 60–74. https://doi.org/10.59573/emsj.9(1).2025.6

Janssens, F. (2011). *On Free Fall in the Three Body Problem* (arXiv:1103.3662). arXiv. https://doi.org/10.48550/arXiv.1103.3662

Krishnaswami, G. S., & Senapati, H. (2019). An introduction to the classical three-body problem: From periodic solutions to instabilities and chaos. *Resonance*, *24*(1), 87–114. https://doi.org/10.1007/s12045-019-0760-1

Kunc, V., & Kléma, J. (2024). *Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks* (arXiv:2402.09092). arXiv. https://doi.org/10.48550/arXiv.2402.09092

Lambert, J. D. (1991). *Numerical methods for ordinary differential systems: The initial value problem*. Wiley.

LIAO, S., Li, X., & Yang, Y. (2021). *Three-body problem—From Newton to supercomputer plus machine learning*. Research Square. https://doi.org/10.21203/rs.3.rs-395522/v1

Liao, S., Li, X., & Yang, Y. (2022). Three-body problem—From Newton to supercomputer plus machine learning. *New Astronomy*, *96*, 101850. https://doi.org/10.1016/j.newast.2022.101850

Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, *63*(1), 208–228. https://doi.org/10.1137/19M1274067

Luo, S. (2020). The Sturm-Liouville problem of two-body system. *Journal of Physics Communications*, *4*(6), 061001. https://doi.org/10.1088/2399-6528/ab9c30

Macukow, B. (2016). Neural Networks – State of Art, Brief History, Basic Models and Architecture. Στο K. Saeed & W. Homenda (Επιμ.), *Lecture Notes in Computer Science: LNCS-9842* (σελ. 3–14). Springer International Publishing. https://doi.org/10.1007/978-3-319-45378-1_1

Marzola, P., Melzer, T., Pavesi, E., Gil-Mohapel, J., & Brocardo, P. S. (2023). Exploring the Role of Neuroplasticity in Development, Aging, and Neurodegeneration. *Brain Sciences*, *13*(12), 1610. https://doi.org/10.3390/brainsci13121610

Montgomery, R. (1998). The N-body problem, the braid group, and action-minimizing periodic solutions. *Nonlinearity*, *11*(2), 363. https://doi.org/10.1088/0951-7715/11/2/011

Montgomery, R. (2014). *The Three-body problem and the shape sphere* (arXiv:1402.0841). arXiv. https://doi.org/10.48550/arXiv.1402.0841

Musielak, Z. E., & Quarles, B. (2014). The three-body problem. *Reports on Progress in Physics*, *77*(6), 065901. https://doi.org/10.1088/0034-4885/77/6/065901

Newton, I. (1687). *Philosophiae Naturalis Principia Mathematica*. Royal Society.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press. http://neuralnetworksanddeeplearning.com/

Pires, P., Santos, J., & Pereira, I. (2023). *Artificial Neural Networks: History and State of the Art*. https://doi.org/10.4018/978-1-6684-7366-5.ch037

Poincaré, H. (1892). *Les Méthodes nouvelles de la mécanique céleste* (τ. 1). Gauthier-Villars.

Roberts, G. E. (2002). Linear Stability of the Elliptic Lagrangian Triangle Solutions in the Three-Body Problem. *Journal of Differential Equations*, *182*(1), 191–218. https://doi.org/10.1006/jdeq.2001.4089

Rowan, C. (2025). *On the failure of ReLU activation for physics-informed machine learning* (arXiv:2512.11184). arXiv. https://doi.org/10.48550/arXiv.2512.11184

Ruder, S. (2017). *An overview of gradient descent optimization algorithms* (arXiv:1609.04747). arXiv. https://doi.org/10.48550/arXiv.1609.04747

Schmidhuber, J. (2022). *Annotated History of Modern AI and Deep Learning* (arXiv:2212.11279). arXiv. https://doi.org/10.48550/arXiv.2212.11279

Šuvakov, M., & Dmitrašinović, V. (2013). Three Classes of Newtonian Three-Body Planar Periodic Orbits. *Physical Review Letters*, *110*(11), 114301. https://doi.org/10.1103/PhysRevLett.110.114301

Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., & Romero-Gonzalez, J. A. (2025). Loss Functions and Metrics in Deep Learning. *Artificial Intelligence*

Review, *58*(7), 195. https://doi.org/10.1007/s10462-025-11198-7

Theodosiou, T., & Rekatsinas, C. (2026). *Physics-Informed Neural Networks without Loss Balancing: A Direct Term Scaling Approach for Nonlinear 1D Problems*. F1000Research. https://doi.org/10.12688/f1000research.169129.2

Wang, H., & Raj, B. (2017). *On the Origin of Deep Learning* (arXiv:1702.07800). arXiv. https://doi.org/10.48550/arXiv.1702.07800

Zhang, J. (2019). *Basic Neural Units of the Brain: Neurons, Synapses and Action Potential* (arXiv:1906.01703). arXiv. https://doi.org/10.48550/arXiv.1906.01703

Zou, Z., Wang, Z., & Karniadakis, G. E. (2025). *Learning and discovering multiple solutions using physics-informed neural networks with random initialization and deep ensemble* (arXiv:2503.06320). arXiv. https://doi.org/10.48550/arXiv.2503.06320

Παράρτημα Α : Αλγόριθμος Mini-batch gradient descent σε γλώσσα python.

```
for i in range(nb_epochs):
    np . random . shuffle ( data )
    for batch in get_batches(data , batch_size =50):
        params_grad = evaluate_gradient(loss_function , batch , params)
        params = params - learning_rate * params_grad
```

Παράρτημα Β : Callback function που δίνει τις αρχικές εξόδους του δικτύου

```
class InitialStateMonitor(dde.callbacks.Callback):
    def __init__(self, t0=0.0):
        super().__init__()
        self.t0 = t0
    def on_epoch_begin(self):
        it = self.model.train_state.iteration
        if it != 0:
            return
        # Evaluate the network at t0
        X0 = np.array([[self.t0]])
        y = self.model.predict(X0)[0]  # shape (8,)
        # Extract positions and velocities
        x1, y1, vx1, vy1 = y[0], y[1], y[2], y[3]
        x2, y2, vx2, vy2 = y[4], y[5], y[6], y[7]
        # Compute inter-body distance
        r12 = np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
        # Print all values with full float64 precision # 17 significant digits captures all float64 precision
        print(f"[iter {it}] Full PINN state at t={self.t0}:")
        print(f"Body 1: x1={x1:.17g}, y1={y1:.17g}, vx1={vx1:.17g}, vy1={vy1:.17g}")
        print(f"Body 2: x2={x2:.17g}, y2={y2:.17g}, vx2={vx2:.17g}, vy2={vy2:.17g}")
        print(f"Inter-body distance r12 = {r12:.17g}")
```

Παράρτημα Γ : AdamW optimizer από την βιβλιοθήκη tensorflow

Η βιβλιοθήκη deepxde έχει backend την βιβλιοθήκη tensorflow και μπορούμε να καλέσουμε optimizers από την tensorflow εφόσον δεν έχουμε υλοποιηθεί ακόμα στην deepxde. Ο κώδικας για την χρησιμοποίηση του adamW δίνεται παρακάτω:

```
import tensorflow as tf

optimizer=tf.keras.optimizers.AdamW(1e-4, weight_decay=1e-4)
```

Παράρτημα Δ : Αλγόριθμος RAR (Residual Adaptive Refinement)

```
 # ============================================================
# RAR LOOP (Residual-based Adaptive Refinement)
# ============================================================

print("\nStarting RAR refinement...\n")

rar_iterations = 21
candidate_points = 10000
points_to_add = 64

for i in range(rar_iterations):

    print(f"\nRAR iteration {i+1}/{rar_iterations}")
    # Sample random time points
    X = geom.random_points(candidate_points)   # (N,1)



    # Compute residual -this costs a lot!
    f = model.predict(X, operator=three_body_ode_second)



    # If operator returns list, stack it
    if isinstance(f, list):
      f = np.hstack(f)   # shape becomes (N, 6)
    # Ensure residual is 1D
    residual = np.mean(np.abs(f), axis=1)

# plot histograms every 2 iterations
    if i%5==0:
      plt.figure(figsize=(6,4))
      plt.hist(residual, bins=100)
      plt.title("Residual Distribution")
      plt.xlabel("Residual")
      plt.ylabel("Count")
      plt.yscale("log")  # important!
      plt.show()
```

```
# Select worst points
  idx = np.argsort(residual)[-points_to_add:]
  X_new = X[idx]

  ###############################################################

  # Force correct shape (M,1)
  X_new = X_new.reshape(-1, 1)

  # Add anchors
  data.add_anchors(X_new)

  # Retrain

  if i <20:  # use only if need to change the learning rate
    lr=1e-4
  else:
    lr=1e-5
  model.compile("adam", lr=lr, loss_weights=loss_weights, loss="MSE")
  model.train(iterations=10000,disregard_previous_best=True,)
```

Παράρτημα Ε: Παραλλαγή αλγόριθμου RAR

```
 # ============================================================
# Enhanced RAR  (Residual-based Adaptive Refinement)
# ============================================================

print("\nStarting RAR refinement...\n")

rar_iterations = 20
candidate_points = 10000
points_to_add = 64

for i in range(rar_iterations):

    print(f"\nRAR iteration {i+1}/{rar_iterations}")
    # Sample random time points
    X = geom.random_points(candidate_points)   # (N,1)



    # Compute residual -this costs a lot!
    f = model.predict(X, operator=three_body_ode_second)



    # If operator returns list, stack it
    if isinstance(f, list):
      f = np.hstack(f)   # shape becomes (N, 6)
    # Ensure residual is 1D
    residual = np.mean(np.abs(f), axis=1)

# plot histograms every 2 iterations
    if i%2==0:
      plt.figure(figsize=(6,4))
      plt.hist(residual, bins=100)
      plt.title("Residual Distribution")
      plt.xlabel("Residual")
      plt.ylabel("Count")
      plt.yscale("log")  # important!
      plt.show()
```

```
# Select worst points

  ###### take the 60%-95% of worst points -- not the worst!
  threshold_low  = np.percentile(residual, 60)
  threshold_high = np.percentile(residual, 95)
  mask = (residual >= threshold_low) & (residual <= threshold_high)
  candidates = X[mask]

  if len(candidates) >= points_to_add:
    idx = np.random.choice(len(candidates), points_to_add, replace=False)
    X_new = candidates[idx]
  else:
  # fallback to top-k
    print("==========> fall back...")
    idx = np.argsort(residual)[-points_to_add:]
    X_new = X[idx]

  ################################################################

  # Force correct shape (M,1)
  X_new = X_new.reshape(-1, 1)



  # Add anchors
  data.add_anchors(X_new)

  # Retrain

  if i <21:  # use only if need to change the learning rate
    lr=1e-4
  else:
    lr=1e-5
  model.compile("adam", lr=lr, loss_weights=loss_weights, loss="MSE")
  model.train(iterations=10000,disregard_previous_best=True,)
```

Παράρτημα Ζ: Υπολογισμός παραγώγων και φασικά διαγράμματα

```
 # ============================================================
# PHASE DIAGRAMS (PINN vs Numerical) using autodiff velocities
# ============================================================
#RESTORE endTime
endTime=1
t_test = np.linspace(0, endTime, 5000)[:, None]
# --- compute PINN positions + velocities with autodiff ---
t_tf = tf.convert_to_tensor(t_test)
with tf.GradientTape() as tape:
    tape.watch(t_tf)
    y_pred = model.net(t_tf)
# ============================================================
# PHASE DIAGRAMS (PINN vs Numerical) using autodiff velocities
# ============================================================
t_tf = tf.convert_to_tensor(t_test)  #time domain points e.g  generated with linspace
with tf.GradientTape() as tape:
    tape.watch(t_tf)
    y_pred = model.net(t_tf)        #forward pass- model predicts the outputs
# Jacobian gives derivatives of each output wrt time
dy_dt = tape.batch_jacobian(y_pred, t_tf)
# remove singleton dimension
dy_dt = dy_dt[:,:,0]
y_pred = y_pred.numpy()
dy_dt = dy_dt.numpy()
# PINN positions
x1, y1 = y_pred[:,0], y_pred[:,1]
x2, y2 = y_pred[:,2], y_pred[:,3]
x3, y3 = y_pred[:,4], y_pred[:,5]
```

```
 # PINN velocities
vx1_p, vy1_p = dy_dt[:,0], dy_dt[:,1]
vx2_p, vy2_p = dy_dt[:,2], dy_dt[:,3]
vx3_p, vy3_p = dy_dt[:,4], dy_dt[:,5]
# ============================================================
# Numerical Solution via SciPy
# ============================================================
def three_body_numeric(t, y, G=1.0, m=(1.0, 1.0, 1.0)):
    x1, y1, vx1, vy1, x2, y2, vx2, vy2, x3, y3, vx3, vy3 = y
    r12 = np.sqrt((x1 - x2)**2 + (y1 - y2)**2 + 1e-6)
    r13 = np.sqrt((x1 - x3)**2 + (y1 - y3)**2 + 1e-6)
    r23 = np.sqrt((x2 - x3)**2 + (y2 - y3)**2 + 1e-6)
    ax1 = G * (m[1]*(x2 - x1)/r12**3 + m[2]*(x3 - x1)/r13**3)
    ay1 = G * (m[1]*(y2 - y1)/r12**3 + m[2]*(y3 - y1)/r13**3)
    ax2 = G * (m[0]*(x1 - x2)/r12**3 + m[2]*(x3 - x2)/r23**3)
    ay2 = G * (m[0]*(y1 - y2)/r12**3 + m[2]*(y3 - y2)/r23**3)
    ax3 = G * (m[0]*(x1 - x3)/r13**3 + m[1]*(x2 - x3)/r23**3)
    ay3 = G * (m[0]*(y1 - y3)/r13**3 + m[1]*(y2 - y3)/r23**3)

    return [vx1, vy1, ax1, ay1,
            vx2, vy2, ax2, ay2,
            vx3, vy3, ax3, ay3]
t_span = (0, endTime)
t_eval = np.linspace(0, endTime, 5000)
#the initial conditions-already defined at the script
y0_full = np.array([
    x1_0, y1_0, vx1_0, vy1_0,
    x2_0, y2_0, vx2_0, vy2_0,
    x3_0, y3_0, vx3_0, vy3_0
])
sol = solve_ivp(three_body_numeric, t_span, y0_full, t_eval=t_eval, rtol=1e-10, atol=1e-12)
x1_n, y1_n = sol.y[0], sol.y[1]
x2_n, y2_n = sol.y[4], sol.y[5]
x3_n, y3_n = sol.y[8], sol.y[9]
# --- numerical velocities ---
vx1_n, vy1_n = sol.y[2], sol.y[3]
vx2_n, vy2_n = sol.y[6], sol.y[7]
vx3_n, vy3_n = sol.y[10], sol.y[11]
```

```
# ============================================================
# PHASE PLOTS
# ============================================================
fig, axes = plt.subplots(3,2, figsize=(12,10))
axes[0,0].plot(x1, vx1_p, 'k--',lw=3, label="PINN")
axes[0,0].plot(x1_n, vx1_n, 'r', label="Numerical")
axes[0,0].scatter(x1_n[0], vx1_n[0], color='r', s=60, label="Start")
axes[0,0].set_title("Body 1: x vs dx/dt")
axes[0,0].grid()
axes[0,1].plot(y1, vy1_p, 'k--',lw=3)
axes[0,1].plot(y1_n, vy1_n, 'r')
axes[0,1].scatter(y1_n[0], vy1_n[0], color='r', s=60, label="Start")
axes[0,1].set_title("Body 1: y vs dy/dt")
axes[0,1].grid()
axes[1,0].plot(x2, vx2_p, 'k--',lw=3)
axes[1,0].plot(x2_n, vx2_n, 'g')
axes[1,0].scatter(x2_n[0], vx2_n[0], color='g', s=60, label="Start")
axes[1,0].set_title("Body 2: x vs dx/dt")
axes[1,0].grid()
axes[1,1].plot(y2, vy2_p, 'k--',lw=3)
axes[1,1].plot(y2_n, vy2_n, 'g')
axes[1,1].scatter(y2_n[0], vy2_n[0], color='g', s=60, label="Start")
axes[1,1].set_title("Body 2: y vs dy/dt")
axes[1,1].grid()
axes[2,0].plot(x3, vx3_p, 'k--',lw=3)
axes[2,0].plot(x3_n, vx3_n, 'b')
axes[2,0].scatter(x3_n[0], vx3_n[0], color='b', s=60, label="Start")
axes[2,0].set_title("Body 3: x vs dx/dt")
axes[2,0].grid()
axes[2,1].plot(y3, vy3_p, 'k--',lw=3)
axes[2,1].plot(y3_n, vy3_n, 'b')
axes[2,1].scatter(y3_n[0], vy3_n[0], color='b', s=60, label="Start")
axes[2,1].set_title("Body 3: y vs dy/dt")
axes[2,1].grid()
plt.tight_layout()
plt.show()
```

Παράρτημα Η: Αλλαγή αρχικών πολώσεων στρώματος εξόδου

Μετά την εντολή model.compile(..) του μοντέλου μπορούμε να αλλάξουμε τα biases με τον παρακάτω κώδικα. Το τελευταίο στρώμα έχει αριθμητική τιμή ίση με το πλήθος των κρυφών στρωμάτων διότι η αρίθμηση των στρωμάτων ξεκινά από το 0.

```
 ################################################################
# Change bias in last layer to avoid xero outputs at t=0
################################################################
model.predict([[0.0]])   #dummy presict to create model.. tensorflow behaviour
# modify bias
last_layer=3                # dense_3 because numbering is:  dense_0 ,1,2,3
model.net.summary()
last_layer = model.net.layers[last_layer]
w, b = last_layer.get_weights()
print("old bias:", b)
b = np.array([-0.3, -0.2, 0, 0, 0.1, 0.4])   # example separation
last_layer.set_weights([w, b])
print("new bias:", b)
################################################################
################################################################
```

Παράρτημα Θ: Μέθοδος ελαχίστων τετραγώνων για εύρεση αρχικών συνθηκών περιοδικών τροχιών.

```
import numpy as np
from scipy.integrate import solve_ivp
from scipy.optimize import least_squares

# -----------------------
# Three-body equations
# -----------------------
def three_body(t, y):
    # y = [x1,y1,x2,y2,x3,y3,vx1,vy1,vx2,vy2,vx3,vy3]
    dydt = np.zeros_like(y)

    # positions
    r = y[:6].reshape(3,2)
    v = y[6:].reshape(3,2)

    # derivatives of position = velocity
    dydt[:6] = v.flatten()

    # accelerations
    a = np.zeros((3,2))
    for i in range(3):
        for j in range(3):
            if i != j:
                diff = r[j] - r[i]
                dist = np.linalg.norm(diff)
                a[i] += diff / dist**3

    dydt[6:] = a.flatten()
    return dydt
```

```
# ----------------------
# Pinn initial conditions
# ----------------------
# x1,y1
# x2,y2
# x3 ,y3
# vx1,vy1
# vx2,vy2
# vx3,vy3
y0 = np.array([
 -0.02060754579306967 ,  0.2062820548044174  ,
0.26842952279231386 ,  0.09176782702268939  ,
-0.22364327446127136 ,  -0.301289552725747  ,

 0.1712777191418524  , -1.6498101110265813,
  0.9394626194134084 ,  0.6423382272799922,
 -1.1534225119266874 ,  1.0107518821541785,
])

T_guess =1.5  # IMPORTANT TO BE CLOSE TO REAL PERIOD!

# ----------------------
# Error function
# ----------------------
def shooting_error(vars):
    # vars = [y0_adjusted..., T]
    y0_new = vars[:-1]
    T = vars[-1]

    sol = solve_ivp(three_body, [0, T], y0_new, rtol=1e-9, atol=1e-9)

    yT = sol.y[:, -1]

    # difference between final and initial state
    return yT - y0_new
```

```
# -----------------------
# Initial guess
# -----------------------
vars0 = np.concatenate([y0, [T_guess]])

# -----------------------
# Optimization
# -----------------------
result = least_squares(shooting_error, vars0, verbose=2, xtol=1e-10)

y0_refined = result.x[:-1]
T_refined = result.x[-1]

print("Refined period:", T_refined)
print("Error norm:", np.linalg.norm(shooting_error(result.x)))

print("\nRefined initial conditions:\n")

labels = [
    "x1","y1","x2","y2","x3","y3",
    "vx1","vy1","vx2","vy2","vx3","vy3"
]

for name, val in zip(labels, y0_refined):
    print(f"{name:>4} = {val:.16f}")
```