ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Εφαρμογή της Μεθόδου Runge-Kutta στα Νευρωνικά Δίκτυα

του

Σωτήριου Λουκά Καμπύλη – ΑΕΜ: 3805

Εκπόνηση πτυχιακής ως μέρος του

Προπτυχιακού Τίτλου Σπουδών

στη

Σχολή Θετικών Επιστημών

Τμήμα Πληροφορικής

Επιβλέπων Καθηγητής: Νικόλαος Τσίτσας

Μάρτιος 2025

Δήλωση Συγγραφικής Ιδιότητας

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στο πλαίσιο αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ευχαριστίες

Η ολοκλήρωση αυτής της πτυχιακής εργασίας δεν θα ήταν δυνατή χωρίς την αμέριστη υποστήριξη και τη βοήθεια πολλών ανθρώπων, τους οποίους θα ήθελα να ευχαριστήσω θερμά.

Πρώτα απ' όλα, θα ήθελα να εκφράσω τις ευχαριστίες μου στην οικογένεια μου. Η αγάπη, η κατανόηση και η συνεχής στήριξη σας ήταν αυτή που με κράτησε στον δρόμο μου και με βοήθησε να ξεπεράσω κάθε δυσκολία. Είστε η πηγή έμπνευσης μου και η κινητήρια δύναμη πίσω από κάθε επιτυχία μου.

Επίσης, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Νικόλαο Τσίτσα, για την καθοδήγηση, την υπομονή και τις πολύτιμες συμβουλές του καθ' όλη τη διάρκεια της εκπόνησης αυτής της εργασίας. Η γνώση και η εμπειρία του ήταν ανεκτίμητες για την ολοκλήρωση αυτής της πτυχιακής.

Μάλιστα, ένα ιδιαίτερο ευχαριστώ απευθύνω στους φίλους μου, που ήταν δίπλα μου σε κάθε βήμα της ακαδημαϊκής μου πορείας. Οι συζητήσεις μας, οι ιδέες και η ηθική υποστήριξη τους ήταν πολύτιμες για την επίτευξη του στόχου μου. Είναι πραγματικά ευτυχές να έχει κανείς δίπλα του ανθρώπους που τον ενθαρρύνουν και τον βοηθούν να αντέξει στις δύσκολες στιγμές.

Ακόμα, θα ήθελα να ευχαριστήσω τους καθηγητές και τους συμφοιτητές μου για τις χρήσιμες συζητήσεις, τις παρατηρήσεις και τις συμβουλές που μου προσέφεραν. Οι γνώσεις και οι εμπειρίες που μοιράστηκαν μαζί μου συνέβαλαν σημαντικά στη βελτίωση της ποιότητας της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω όλους όσους με τον έναν ή τον άλλο τρόπο συνέβαλαν στην πραγματοποίηση αυτής της πτυχιακής. Η εργασία αυτή είναι αποτέλεσμα συλλογικής προσπάθειας και συνεργασίας όλων των παραπάνω.

Περίληψη

Η παρούσα πτυχιακή εργασία εξετάζει την εφαρμογή της μεθόδου Runge-Kutta (RK4) στα νευρωνικά δίκτυα, με σκοπό τη μείωση του χρόνου εκπαίδευσης, τη βελτίωση της απόδοσης και την ενίσχυση των ικανοτήτων γενίκευσης τους. Συγκεκριμένα, η μέθοδος αυτή χρησιμοποιείται για την επίλυση Προβλημάτων Αρχικών Τιμών (ΠΑΤ) που σχετίζονται με Συνήθεις Διαφορικές Εξισώσεις (ΣΔΕ), αλλά δείχνει θετικά αποτελέσματα στην βελτίωση των νευρωνικών δικτύων για την επίλυση προβλημάτων στον πραγματικό κόσμο. Με άλλα λόγια, η βασική ιδέα είναι ότι μπορεί να προσφέρει υψηλή ακρίβεια στις προβλέψεις και να αποτρέψει τα συνήθη προβλήματα που προκύπτουν στα νευρωνικά δίκτυα λόγω περίπλοκων υπολογισμών.

Δυστυχώς, γνωρίζουμε πως οι παραδοσιακές μέθοδοι για την εκπαίδευση και τη βελτιστοποίηση των νευρωνικών δικτύων συνήθως δεν επαρκούν, ιδιαίτερα σε σύνθετα συστήματα. Συνεπώς, η έρευνα στον τομέα αυτό στοχεύει να δείξει πώς η ενσωμάτωση της μεθόδου Runge-Kutta στα νευρωνικά δίκτυα μπορεί να παρέχει λύσεις σε διάφορα προβλήματα, προσφέροντας ταυτόχρονα καλύτερη γενίκευση και πιο αξιόπιστες προβλέψεις.

Αρχικά, εξηγούνται σε βάθος τα Νευρωνικά Δίκτυα, μαζί με τις Συνήθεις Διαφορικές Εξισώσεις, τα Προβλήματα Αρχικών Τιμών και τη μέθοδο Runge-Kutta, καθώς αυτά είναι κρίσιμα για την κατανόηση του προβλήματος που εξετάζεται. Ακολούθως, παρέχεται μια επισκόπηση των παραδοσιακών τεχνικών βελτιστοποίησης για τα νευρωνικά δίκτυα αλλά και των προκλήσεων που αντιμετωπίζουν. Δηλαδή, στηριζόμαστε σε διεθνή βιβλιογραφία που η εφαρμογή της μεθόδου Runge-Kutta στα νευρωνικά δίκτυα οδηγεί σε βελτιωμένες προβλέψεις και αν τα αποτελέσματα που προκύπτουν από τα RKNNs (Runge-Kutta Neural Networks) είναι ανώτερα σε σύγκριση με άλλους τύπους νευρωνικών δικτύων.

Τέλος, η εργασία ασχολείται με την πρακτική εφαρμογή της μεθόδου Runge-Kutta σε νευρωνικά δίκτυα που επιλύουν ΠΑΤ με ΣΔΕ. Ουσιαστικά, διεξήχθησαν διάφορα πειράματα χρησιμοποιώντας κώδικα σε Python. Με αυτόν τον τρόπο, παρουσιάζεται πώς μπορεί να χρησιμοποιηθεί η μέθοδος για την ενίσχυση της αποτελεσματικότητας κατά τη διαδικασία εκπαίδευσης, βελτιώνοντας την πρόβλεψη. Υπό αυτήν την έννοια, τα ερευνητικά αποτελέσματα που επιτυγχάνονται σε αυτή την εργασία προσφέρουν περαιτέρω πληροφορίες σχετικά με την σύνδεση αυτών των δύο εννοιών, ως βάση για μελλοντικές συνεισφορές.

Λέξεις Κλειδιά: Μέθοδος Runge-Kutta (RK4), Νευρωνικά Δίκτυα, Runge-Kutta Νευρωνικό Δίκτυο (RKNN), Συνήθεις Διαφορικές Εξισώσεις (ΣΔΕ), Προβλήματα Αρχικών Τιμών (ΠΑΤ), Βελτιστοποίηση.

Abstract

This thesis explores the application of the Runge-Kutta method (RK4) in neural networks, aiming to reduce training time, enhance performance, and improve generalization capabilities. Traditionally used to solve Initial Value Problems (IVPs) associated with Ordinary Differential Equations (ODEs), the RK4 method has demonstrated promising results in enhancing neural networks for real world problem solving. Essentially, its key advantage lies in providing high prediction accuracy while mitigating common issues in neural networks caused by complex computations.

Conventional training and optimization methods for neural networks often fall short, particularly when dealing with complex systems. Therefore, this research aims to demonstrate how integrating the Runge-Kutta method into neural networks can address various challenges, offering better generalization and more reliable predictions.

The thesis begins with an in-depth explanation of Neural Networks, Ordinary Differential Equations, Initial Value Problems, and the Runge-Kutta method, laying the theoretical foundation for the study. This is followed by a comprehensive review of traditional optimization techniques and the challenges they present. Drawing on international research, the study evaluates whether applying the Runge-Kutta method to neural networks can lead to improved predictive performance and whether the results from Runge-Kutta Neural Networks (RKNNs) surpass those of other neural network architectures.

Finally, the practical application of the Runge-Kutta method in neural networks solving IVPs with ODEs is examined. Various experiments, implemented in Python, showcase how the RK4 method enhances training efficiency and improve predictive accuracy. The findings of this research contribute valuable insights into the intersection of these two fields, laying the groundwork for future developments.

Keywords: Runge-Kutta Method (RK4), Neural Networks, Runge-Kutta Neural Network (RKNN), Ordinary Differential Equations (ODEs), Initial Value Problems (IVPs), Optimization.

Περιεχόμενα

1 Εισαγωγή και Σκοπός της Μελέτης

1.1 Εισαγωγή

Τα νευρωνικά δίκτυα συγκαταλέγονται στα πιο σημαντικά και ταχύτερα αναπτυσσόμενα πεδία έρευνας στην τεχνητή νοημοσύνη. Ωστόσο, στη διαδικασία εκπαίδευσης εμπλέκονται σύνθετες μαθηματικές συναρτήσεις, όπου οι παραδοσιακές τεχνικές βελτιστοποίησης, όπως ο Gradient Descent και ο Adam, παρουσιάζουν προβλήματα σύγκλισης και τοπικών ελαχίστων. Έτσι, η αναζήτηση εναλλακτικών μεθόδων για τη βελτίωση της απόδοσης των νευρωνικών δικτύων έχει εξελιχθεί σε ένα από τα μεγαλύτερα ερευνητικά πεδία, με τον αριθμό των ερευνητών να αυξάνεται καθημερινά.

Επιπλέον, διάφορες παραλλαγές της μεθόδου Runge-Kutta έχουν εφαρμοστεί στην αριθμητική ανάλυση και έχουν αποδειχθεί ιδιαίτερα αποδοτικές για μια ευρεία κατηγορία προβλημάτων. Μεταξύ αυτών, ίσως η πιο γνωστή μορφή της μεθόδου Runge-Kutta είναι η RK4, ένας αριθμητικός τρόπος επίλυσης συστημάτων συνήθων διαφορικών εξισώσεων, που αποτέλεσε εξαιρετικά δημοφιλή προσέγγιση για Προβλήματα Αρχικών Τιμών (ΠΑΤ) με Συνήθεις Διαφορικές Εξισώσεις (ΣΔΕ). Αν και δεν χρησιμοποιείται παραδοσιακά για την εκπαίδευση νευρωνικών δικτύων, η ικανότητα της να παρέχει ακριβείς προσεγγίσεις σε συστήματα που περιλαμβάνουν διαφορικές εξισώσεις υποδεικνύει έντονα ότι είναι μια πολλά υποσχόμενη τεχνική για την εφαρμογή στη βελτιστοποίηση των νευρωνικών δικτύων.

Η διεθνής βιβλιογραφία επίσης υποδεικνύει ότι η χρήση της μεθόδου Runge-Kutta μπορεί να συμβάλει στην επιτάχυνση της σύγκλισης των νευρωνικών δικτύων, αποφεύγοντας έτσι καθυστερήσεις και άλλα προβλήματα που συνδέονται με τη σύγκλιση σε τοπικά ελάχιστα. Μία ιδιαίτερα ενδιαφέρουσα εργασία [1] δείχνει ότι η γενίκευση και η ικανότητα μακροχρόνιας πρόβλεψης των Runge-Kutta Νευρωνικών Δικτύων υπερτερεί άλλων παραλλαγών νευρωνικών δικτύων.

Αυτό τελικά σημαίνει ότι η εφαρμογή της μεθόδου Runge-Kutta για την εκπαίδευση νευρωνικών δικτύων παρουσιάζει μεγάλο ερευνητικό ενδιαφέρον και τεράστιες δυνατότητες για τη βελτίωση της απόδοσης και της αποτελεσματικότητας των νευρωνικών δικτύων. Επομένως, η μελέτη και η εφαρμογή αυτής της μεθοδολογίας σε πραγματικά προβλήματα μάθησης θα ανοίξει νέες προσεγγίσεις για την εκπαίδευση και την ανάπτυξη πιο ανθεκτικών συστημάτων τεχνητής νοημοσύνης.

1.2 Σκοπός της Πτυχιακής Εργασίας

Ο βασικός σκοπός της παρούσας πτυχιακής εργασίας είναι να εξετάσει την εφαρμογή της μεθόδου Runge-Kutta (RK4), τόσο θεωρητικά όσο και πρακτικά, στη βελτιστοποίηση των νευρωνικών δικτύων. Συνοπτικά, ο κύριος στόχος είναι να διαπιστωθεί αν αυτή η αριθμητική μέθοδος μπορεί να συμβάλει στη βελτίωση της απόδοσης των νευρωνικών δικτύων κατά τη διαδικασία εκπαίδευσης τους και στα συμπεράσματα που προκύπτουν από αυτά. Συγκεκριμένα, η μέθοδος Runge-Kutta, η οποία παραδοσιακά χρησιμοποιείται για την επίλυση διαφορικών εξισώσεων, θα εξεταστεί ως εργαλείο προσαρμογής των παραμέτρων των νευρωνικών δικτύων, με έμφαση στην ταχύτητα σύγκλισης και την αποφυγή τοπικών ελαχίστων κατά την εκπαίδευση. Πράγματι, η εισαγωγή αυτής της μεθόδου αναμένεται να επιτρέψει την ανάπτυξη πιο αποδοτικών τεχνικών βελτιστοποίησης για τα νευρωνικά δίκτυα, επιταχύνοντας τη διαδικασία εκπαίδευσης και μειώνοντας τα προβλήματα που σχετίζονται με τη σύγκλιση σε τοπικά ελάχιστα.

Αρχικά, η κινητήρια δύναμη πίσω από αυτή την πτυχιακή εργασία έγκειται στο γεγονός ότι η βελτιστοποίηση των νευρωνικών δικτύων αποτελεί μια από τις πιο κρίσιμες προκλήσεις στην ανάπτυξη σύγχρονων αλγορίθμων τεχνητής νοημοσύνης. Αν και οι παραδοσιακοί αλγόριθμοι βελτιστοποίησης (όπως οι Gradient Descent και Adam) είναι πρακτικοί και αποτελεσματικοί, συχνά αντιμετωπίζουν προβλήματα σχετικά με την ταχύτητα σύγκλισης και την αποφυγή τοπικών ελαχίστων. Αντίθετα, η μέθοδος Runge-Kutta, που χρησιμοποιείται κυρίως σε άλλους τομείς, μπορεί να προσφέρει μια καινοτόμο προσέγγιση στη βελτιστοποίηση των νευρωνικών δικτύων, αντιμετωπίζοντας ορισμένες από τις αδυναμίες των παραδοσιακών μεθόδων. Ειδικότερα, ο συνδυασμός αριθμητικής ακρίβειας και ευελιξίας της μεθόδου Runge-Kutta μπορεί να προσφέρει μια πιο αξιόπιστη και ταχύτερη διαδικασία εκπαίδευσης, κάτι που θα έχει θετικό αντίκτυπο στην αποδοτικότητα των νευρωνικών δικτύων σε πολλούς τομείς εφαρμογής.

Τέλος, η εφαρμογή αυτής της μεθόδου στα νευρωνικά δίκτυα αποτελεί έναν καινοτόμο τομέα έρευνας που συνδυάζει τη θεωρία των διαφορικών εξισώσεων με τις σύγχρονες τεχνικές μηχανικής μάθησης. Συμπερασματικά, η παρούσα μελέτη αποσκοπεί να αναδείξει τις δυνατότητες της μεθόδου Runge-Kutta στην ανάπτυξη πιο ισχυρών και αποδοτικών αλγορίθμων και να προσφέρει νέες προοπτικές για τη βελτιστοποίηση της εκπαίδευσης των νευρωνικών δικτύων σε πολύπλοκα συστήματα.

1.3 Διάρθρωση Πτυχιακής Εργασίας

Η παρούσα πτυχιακή εργασία οργανώνεται σε έξι κύρια κεφάλαια, τα οποία καλύπτουν διεξοδικά τις θεωρητικές και πρακτικές πτυχές της εφαρμογής της μεθόδου Runge-Kutta στα νευρωνικά δίκτυα. Κάθε κεφάλαιο έχει σχεδιαστεί για να συνεισφέρει στη συνολική κατανόηση του αντικειμένου, ξεκινώντας από τις βασικές θεωρητικές έννοιες και καταλήγοντας στην πειραματική εφαρμογή και τα συμπεράσματα.

- **Κεφάλαιο 1** (*Εισαγωγή και Σκοπός της Μελέτης*): Το πρώτο κεφάλαιο παρουσιάζει το γενικό πλαίσιο της εργασίας. Αναδεικνύεται η σημασία των νευρωνικών δικτύων και η ανάγκη για νέες μεθόδους βελτιστοποίησης, με έμφαση στη μέθοδο Runge-Kutta. Παρουσιάζεται ο σκοπός και η συμβολή της μελέτης, καθώς και οι κύριοι στόχοι που επιδιώκει να επιτύχει.
- **Κεφάλαιο 2** (*Θεωρητικές Βάσεις Νευρωνικών Δικτύων*): Σε αυτό το κεφάλαιο αναπτύσσονται οι βασικές θεωρητικές αρχές που διέπουν τα νευρωνικά δίκτυα, καθώς και οι συναφείς έννοιες της τεχνητής νοημοσύνης, της μηχανικής και της βαθιάς μάθησης. Παρουσιάζονται οι κύριοι τύποι νευρωνικών δικτύων (MLP, CNN, RNN, LSTM) και οι αλγόριθμοι βελτιστοποίησης που χρησιμοποιούνται για την εκπαίδευση τους.
- **Κεφάλαιο 3** (*Αριθμητική Ανάλυση και Μέθοδοι Runge-Kutta*): Το τρίτο κεφάλαιο επικεντρώνεται στις αριθμητικές μεθόδους για την επίλυση ΣΔΕ. Παρουσιάζεται η θεωρία των ΣΔΕ και τα ΠΑΤ, ενώ παράλληλα αναλύονται οι διαφορετικές τάξεις της μεθόδου Runge-Kutta (1ης, 2ης, 3ης και 4ης τάξης), με έμφαση στη μέθοδο RK4.
- **Κεφάλαιο 4** (*Σχετική Βιβλιογραφία*): Σε αυτό το κεφάλαιο γίνεται ανασκόπηση της υπάρχουσας βιβλιογραφίας σχετικά με την εφαρμογή της μεθόδου Runge-Kutta στα νευρωνικά δίκτυα. Αναλύονται οι σχετικές μελέτες και τα ερευνητικά ευρήματα, με στόχο να τεκμηριωθεί θεωρητικά η σκοπιμότητα της προτεινόμενης προσέγγισης.
- **Κεφάλαιο 5** (*Πειραματισμός και Αποτελέσματα*): Το πέμπτο κεφάλαιο περιγράφει την πειραματική διαδικασία και παρουσιάζει τα αποτελέσματα. Συγκεκριμένα, συγκρίνονται οι επιδόσεις των διαφόρων εκδοχών της μεθόδου Runge-Kutta (RK1, RK2, RK3, RK4) και έπειτα ένας custom RK4 βελτιστοποιητής με τους παραδοσιακούς βελτιστοποιητές όπως ο Gradient Descent και ο Adam. Τα πειράματα διεξήχθησαν σε Python και αξιολογήθηκε η αποδοτικότητα των αλγορίθμων στην επίλυση προβλημάτων ΠΑΤ με ΣΔΕ.
- **Κεφάλαιο 6** (*Συμπεράσματα και Μελλοντικές Επεκτάσεις*): Το τελευταίο κεφάλαιο συνοψίζει τα αποτελέσματα της έρευνας, αναλύει τα βασικά

ευρήματα και αξιολογεί τη συμβολή της μεθόδου Runge-Kutta στη βελτιστοποίηση των νευρωνικών δικτύων. Επιπλέον, προτείνονται κατευθύνσεις για μελλοντική έρευνα και περαιτέρω ανάπτυξη της μεθοδολογίας.

Η δομή της εργασίας έχει σχεδιαστεί έτσι ώστε να παρέχει μια πλήρη και ολοκληρωμένη παρουσίαση του θέματος, συνδυάζοντας τη θεωρητική ανάλυση, την πειραματική διερεύνηση αλλά και τη διατύπωση συμπερασμάτων, ενώ παράλληλα προτείνει κατευθύνσεις για μελλοντική έρευνα και ανάπτυξη.

2 Θεωρητικές Βάσεις Νευρωνικών Δικτύων

2.1 Τεχνητή Νοημοσύνη

Η Τεχνητή Νοημοσύνη, ή AI εν συντομία, είναι μια επιστήμη που ασχολείται με την ανάπτυξη αλγορίθμων και συστημάτων που μιμούνται την ανθρώπινη νοημοσύνη. Σύμφωνα με το βιβλίο "*Artificial Intelligence: A Modern Approach, Third Edition*" [2], οι θεωρητικές της βάσεις άρχισαν να διαμορφώνονται με τους Warren McCulloch και Walter Pitts το 1943, βασισμένες κυρίως σε τρεις πηγές: α) την βασική κατανόηση του εγκεφάλου, β) τη τυπική λογική και γ) τη θεωρία υπολογισμού. Ο πρώτος υπολογιστής νευρωνικού δικτύου, SNARC, κατασκευάστηκε το 1950 από τους Marvin Minsky και Dean Edmonds. Ωστόσο, ο Alan Turing αναγνωρίζεται γενικά ως ο κύριος θεμελιωτής της τεχνητής νοημοσύνης. Πράγματι, ο όρος πρωτοχρησιμοποιήθηκε από τον John McCarthy σε ένα δίμηνο σεμινάριο στο Dartmouth College το καλοκαίρι του 1956.

Επιπλέον, η Τεχνητή Νοημοσύνη έχει εξελιχθεί με την ενσωμάτωση γνώσεων από οκτώ επιστημονικούς τομείς, εμπλουτίζοντας έτσι το πεδίο της. Για παράδειγμα, από τη Φιλοσοφία (π.χ. θεωρίες μυαλού και σώματος), Μαθηματικά (π.χ. μέσω τυπικών πλαισίων), Οικονομικά (π.χ. θεωρία απόφασης και θεωρία παιγνίων), Νευροεπιστήμη (π.χ. μέσω EEG), Ψυχολογία (π.χ. μέσω συμπεριφορικών μηχανισμών εισόδου-εξόδου), Μηχανική Υπολογιστών (π.χ. μέσω υλικού και λογισμικού που επιτρέπει τις εφαρμογές AI), Θεωρία Ελέγχου (π.χ. τεχνικές αυτοδιαχείρισης μηχανών χωρίς γλώσσα ή προγραμματισμό, αντίθετα με τους ψηφιακούς υπολογιστές), και Γλωσσολογία (π.χ. υπολογιστική γλωσσολογία ή φυσική επεξεργασία γλώσσας) [3].

Το πεδίο της Τεχνητής Νοημοσύνης περιλαμβάνει βασικές υποκατηγορίες. Μια σημαντική και ταχύτατα αναπτυσσόμενη υποκατηγορία είναι η Μηχανική Μάθηση, η οποία έχει πλέον αναδειχθεί στο επίκεντρο της κατασκευής περίπλοκων συστημάτων. Η Βαθιά Μάθηση είναι μια εξαιρετικά δημοφιλής περιοχή εντός της AI. Χρησιμοποιεί βαθιά νευρωνικά δίκτυα για την αναγνώριση προτύπων, με ολοένα πιο σύνθετα δίκτυα για εικόνες, ήχους και κείμενα. Αυτό έχει οδηγήσει σε επαναστατικές εφαρμογές, όπως η αυτόματη αναγνώριση ομιλίας, η μετάφραση μηχανής και η διάγνωση ασθενειών όπως η νόσος Alzheimer.

Ωστόσο, η ανάπτυξη της Τεχνητής Νοημοσύνης συνοδεύεται από αρκετές προκλήσεις και ηθικά ζητήματα. Συγκεκριμένα, περιλαμβάνει τη διαφάνεια των αλγορίθμων, την προστασία της ιδιωτικότητας και τις πιθανές προκαταλήψεις στα δεδομένα. Επιπλέον, η αυτοματοποίηση μπορεί να επηρεάσει την απασχόληση, απαιτώντας την απόκτηση νέων δεξιοτήτων και την προσαρμογή στα μεταβαλλόμενα εργασιακά πρότυπα [4].

Στο τέλος, αποτελεί ένα πεδίο με μεγάλες υποσχέσεις, καθώς συνεχώς δημιουργούνται νέες μέθοδοι και εφαρμογές. Η διαρκής έρευνα και η

διεπιστημονική συνεργασία έχουν ιδιαίτερη σημασία για την αντιμετώπιση αυτών των προκλήσεων, ενώ η αξιοποίηση των τεχνολογιών της επόμενης γενιάς, όπως η Εξηγήσιμη Τεχνητή Νοημοσύνη, αναμένεται να ενισχύσει ακόμη περισσότερο την αποδοχή και τη χρήση της από την κοινωνία, υπό την προϋπόθεση ότι θα είναι συμβατή με τις ανθρώπινες αξίες και ανάγκες [5].

2.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning - ML) [6] αποτελεί έναν υποκλάδο της τεχνητής νοημοσύνης, ο οποίος περιλαμβάνει αλγόριθμους και μοντέλα που επιτρέπουν στους υπολογιστές να μαθαίνουν και να βελτιώνουν την απόδοσή τους. Αυτό επιτυγχάνεται κυρίως μέσω της εμπειρίας και της ανάλυσης δεδομένων, χωρίς να απαιτείται ρητός προγραμματισμός για κάθε συγκεκριμένη εργασία. Η ML επιτρέπει στους υπολογιστές να αναγνωρίζουν πρότυπα και να αποκτούν γνώσεις, προσαρμόζοντας τις αποφάσεις τους σύμφωνα με τα δεδομένα που επεξεργάζονται. Αντίθετα με τις παραδοσιακές προσεγγίσεις, χρησιμοποιεί δεδομένα για να προβλέπει ή να λαμβάνει αποφάσεις ακόμα και σε περιπτώσεις όπου οι παραδοσιακοί αλγόριθμοι αποτυγχάνουν.

Η ιστορία της Μηχανικής Μάθησης ξεκινά από τον Arthur Samuel, ο οποίος εισήγαγε τον όρο το 1959, περιγράφοντας την ιδέα των ανεξάρτητα μαθησιακών μηχανών. Κατά τη δεκαετία του 1960, το πεδίο προχώρησε με την έρευνα του Nilsson στην ταξινόμηση προτύπων, ενώ τη δεκαετία του 1970 ενέπνευσε άλλους πρωτοπόρους, όπως τους Duda και Hart. Το 1981, μία μελέτη παρουσίασε εκπαιδευτικές μεθόδους που χρησιμοποίησαν νευρωνικά δίκτυα για την αναγνώριση 40 χαρακτήρων. Αργότερα, ο Tom M. Mitchell όρισε τη μηχανική μάθηση ως τη διαδικασία βελτίωσης της απόδοσης ενός προγράμματος υπολογιστή σε μια συγκεκριμένη εργασία μέσω εμπειρίας, με την απόδοση να εκτιμάται μέσω ενός συγκεκριμένου μέτρου.

Η μηχανική μάθηση μπορεί να χωριστεί σε τρεις κύριες κατηγορίες: α) Επιβλεπόμενη Μάθηση, β) Μη Επιβλεπόμενη Μάθηση, και γ) Ενισχυτική Μάθηση. Αρχικά, η επιβλεπόμενη μάθηση περιλαμβάνει αλγόριθμους που δημιουργούν μοντέλα από δεδομένα με ετικέτες, ώστε να προβλέπουν τις εξόδους νέων εισόδων και να τις ταξινομούν σε διακριτές ή συνεχείς τιμές. Στη συνέχεια, η μη επιβλεπόμενη μάθηση χρησιμοποιεί διάφορους αλγόριθμους για να αναγνωρίσει πρότυπα από δεδομένα χωρίς την ανάγκη ετικετοποιημένων δεδομένων, εφαρμόζοντας τεχνικές όπως η ομαδοποίηση και η ανίχνευση ανωμαλιών. Τέλος, η ενισχυτική μάθηση στοχεύει στην εκπαίδευση λογισμικών πρακτόρων που δρουν σε περιβάλλοντα για τη μεγιστοποίηση των ανταμοιβών. Οι τύποι αυτοί θα παρουσιαστούν πιο αναλυτικά στη συνέχεια.

2.2.1 Επιβλεπόμενη Μάθηση

Η επιβλεπόμενη μάθηση [7] είναι μία από τις πιο βασικές και ευρέως χρησιμοποιούμενες τεχνικές στην τεχνητή νοημοσύνη και συνεπώς στη μηχανική μάθηση. Η γενική ιδέα αυτής της μεθόδου βασίζεται στην εκπαίδευση ενός μοντέλου χρησιμοποιώντας δεδομένα που περιλαμβάνουν εισόδους και τις αντίστοιχες εξόδους ή ετικέτες τους, ώστε το σύστημα να μάθει τη συσχέτιση μεταξύ εισόδων και εξόδων.

Βασικά, ολόκληρη η διαδικασία της επιβλεπόμενης μάθησης ξεκινά με τη χρήση ενός συνόλου εκπαίδευσης που περιέχει παραδείγματα με γνωστά αποτελέσματα ή ετικέτες. Το μοντέλο συνήθως εκπαιδεύεται έτσι ώστε να μαθαίνει τις σχέσεις που συνδέουν τις εισόδους με τις εξόδους. Κατά την εκπαίδευση, το μοντέλο συνεχίζει να προσαρμόζει τις παραμέτρους του, συνήθως τα βάρη στα νευρωνικά δίκτυα, ώστε να ελαχιστοποιήσει το σφάλμα μεταξύ των προβλέψεών του και των πραγματικών ετικετών, χρησιμοποιώντας συνήθως μια συνάρτηση απώλειας. Αυτή η διαδικασία συνεχίζεται μέχρι το μοντέλο να επιτύχει καλή απόδοση στο εκπαιδευτικό σύνολο δεδομένων. Μεταξύ των μεθόδων επιβλεπόμενης μάθησης, μερικές είναι ιδιαίτερα αξιοσημείωτες, καθώς είναι γραμμικές: η γραμμική παλινδρόμηση (Linear Regression) και τα δέντρα απόφασης (Decision Trees).

Όταν η έξοδος λαμβάνει ένα πεπερασμένο σύνολο διακριτών τιμών που αντιπροσωπεύουν τις ετικέτες κατηγορίας της εισόδου, τότε η συσχέτιση που μαθαίνεται έχει ως αποτέλεσμα την ταξινόμηση των δεδομένων. Αντίθετα, όταν η έξοδος πρέπει να περιλαμβάνει συνεχόμενες τιμές, η διαδικασία προβλέπει τις τιμές εισόδου μέσω παλινδρόμησης. Συνοπτικά, η επιβλεπόμενη μάθηση παρέχει ποικίλες λύσεις για διαφορετικούς τύπους προβλημάτων, από την ταξινόμηση μέχρι την πρόβλεψη συνεχών τιμών ή παλινδρόμησης. Για παράδειγμα, οι μέθοδοι δέντρων απόφασης έχουν χρησιμοποιηθεί ευρέως σε προβλήματα ταξινόμησης και έχουν αποδώσει καλά σε πραγματικές εφαρμογές, όπως η ανάλυση χρηματιστηριακών αγορών.

Οι εφαρμογές της επιβλεπόμενης μάθησης έχουν επίσης επεκταθεί σε πολλούς άλλους τομείς: στην υγειονομική περίθαλψη χρησιμοποιούνται για την ανάλυση ιατρικών εικόνων και τη διάγνωση, στη χρηματοοικονομική ανάλυση για την πρόβλεψη τιμών μετοχών και τη μοντελοποίηση κινδύνου, στη βιομηχανία αυτοκινήτων για την ανάπτυξη αυτόνομων οχημάτων, ενώ στη ρομποτική βοηθούν στη βελτίωση της αλληλεπίδρασης ανθρώπου-ρομπότ και χρησιμοποιούνται για την ενίσχυση της αυτονομίας των ρομπότ.

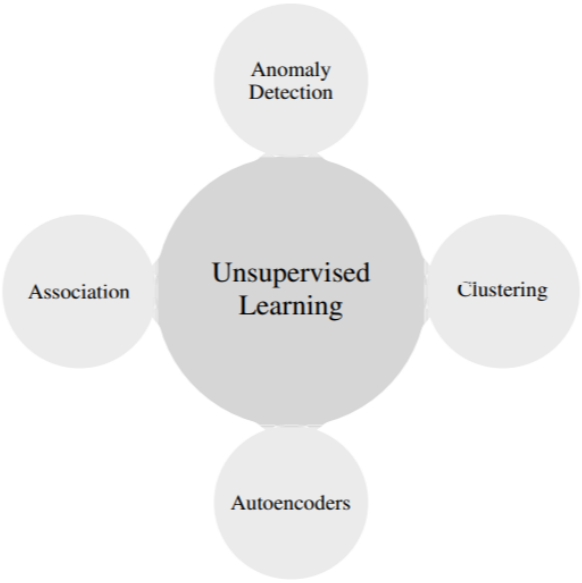
Ωστόσο, παρά την αποτελεσματικότητα της επιβλεπόμενης μάθησης, υπάρχουν ορισμένοι περιορισμοί. Ένα από τα κύρια προβλήματα είναι η ανάγκη για μεγάλες ποσότητες επισημασμένων δεδομένων· η συλλογή τέτοιων δεδομένων μπορεί να είναι δαπανηρή και χρονοβόρα. Επιπλέον, αν δεν χρησιμοποιηθούν κατάλληλες τεχνικές κανονικοποίησης και επικύρωσης, τα μοντέλα επιβλεπόμενης μάθησης είναι επιρρεπή σε υπερεκπαίδευση (overfitting). Η ανάπτυξη τεχνικών

που μειώνουν την εξάρτηση από μεγάλα σύνολα δεδομένων, όπως η μεταφορά μάθησης (transfer learning), μπορεί να αποτελέσει πιθανή λύση σε αυτά τα ζητήματα.

2.2.2 Μη Επιβλεπόμενη Μάθηση

Η Μη Επιβλεπόμενη Μάθηση [8] είναι μια άλλη κύρια κατηγορία της μηχανικής μάθησης, η οποία αφορά την εκπαίδευση μοντέλων χωρίς την απαίτηση για επισημασμένα δεδομένα. Σε αντίθεση με την επιβλεπόμενη μάθηση, όπου οι είσοδοι συνοδεύονται από τις αντίστοιχες ετικέτες, η μη επιβλεπόμενη μάθηση επικεντρώνεται στην ανακάλυψη εγγενών δομών ή προτύπων μέσα στα δεδομένα, χωρίς εξωτερικές επισημάνσεις. Αυτό τη καθιστά εξαιρετικά χρήσιμη για την κατανόηση, την ομαδοποίηση ή την ανάλυση δεδομένων αυτόματα, ιδίως σε προβλήματα όπου η επισήμανση των δεδομένων είναι δύσκολη, δαπανηρή ή περιττή. Επιπλέον, η μέθοδος αυτή είναι κατάλληλη σε περιπτώσεις όπου η κατηγοριοποίηση των δεδομένων δεν είναι γνωστή εκ των προτέρων ή η επίβλεψη από τον χρήστη δεν είναι απαραίτητη. Γενικά, οι βασικές προσεγγίσεις περιλαμβάνουν τύπους μη επιβλεπόμενης μάθησης, όπως:

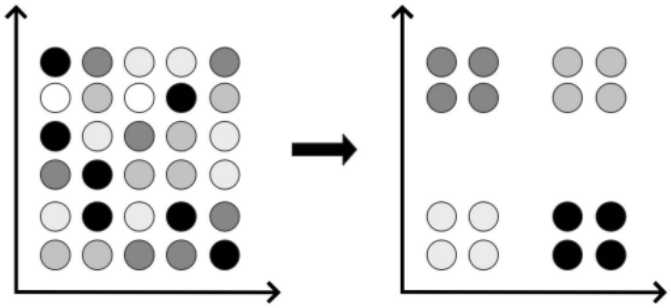
- Ομαδοποίηση (Clustering)
- Συσχέτιση (Association)
- Ανίχνευση Ανωμαλιών (Anomaly Detection)
- Αυτόματοι Κωδικοποιητές (Autoencoders)



Σχήμα 2.1: Τύποι Μη Επιβλεπόμενης Μάθησης [8]

Η πιο συχνή χρήση της μη επιβλεπόμενης μάθησης εντοπίζεται στην Ομαδοποίηση (Clustering), όπου το μοντέλο προσπαθεί να βρει ομάδες ή

υποσύνολα παρόμοιων δεδομένων. Υπάρχουν διάφοροι τύποι ομαδοποίησης, όπως η διαμεριστική, η ιεραρχική, η επικαλυπτόμενη και η πιθανοτική. Ένας από τους πιο γνωστούς αλγορίθμους διαμεριστικής ομαδοποίησης είναι ο K-means, ο οποίος ταξινομεί τα δεδομένα σε Κ ομάδες, βασιζόμενος στη μικρότερη απόσταση από τα κέντρα των ομάδων. Οι εφαρμογές της ομαδοποίησης κυμαίνονται από την αναγνώριση προτύπων έως την ανάλυση συμπεριφοράς καταναλωτών, αξιοποιώντας κατανομές πιθανότητας για τη δημιουργία των ομάδων.



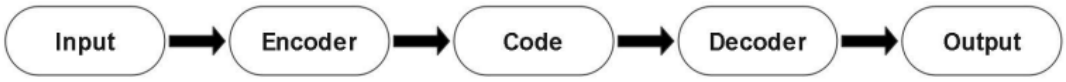
Σχήμα 2.2: Τυχαίο Παράδειγμα Ομαδοποίησης (Clustering) [8]

Η Συστηματική Ανάλυση Συσχετίσεων (Association Rule Learning) είναι μία ακόμη τεχνική ανάλυσης, όπου οι αλγόριθμοι μη επιβλεπόμενης μάθησης χρησιμοποιούνται για τον εντοπισμό συσχετίσεων στα δεδομένα. Ο στόχος εδώ είναι η ανεύρεση κανόνων που περιγράφουν σχέσεις μεταξύ διάφορων χαρακτηριστικών στα δεδομένα, χωρίς την ανάγκη για επισημασμένη εποπτεία. Από την ανίχνευση απλών καταναλωτικών συνηθειών έως την ανάπτυξη πιο σύνθετων στρατηγικών πωλήσεων και στοχευμένης διαφήμισης, αυτές οι τεχνικές βρίσκουν ευρύτατες εφαρμογές στο εμπόριο. Με απλούς όρους, αυτή η μέθοδος μπορεί να θεωρηθεί ως προχωρημένη ανάλυση σεναρίων, όπου η σχέση περιγράφεται ως "αν συμβεί κάτι, τότε ακολουθεί κάτι άλλο."

Εκτός από τις παραδοσιακές τεχνικές, πιο πρόσφατα μοντέλα μη επιβλεπόμενης μάθησης περιλαμβάνουν τους Αυτόματους Κωδικοποιητές (Autoencoders), που είναι νευρωνικά δίκτυα που μαθαίνουν μία αναπαράσταση των δεδομένων μέσω της συμπίεσης και της ανακατασκευής τους. Ένας αυτόματος κωδικοποιητής αποτελείται από τρία κύρια στοιχεία:

1. Κωδικοποιητής: Συμπιέζει την είσοδο σε μία αναπαράσταση χαμηλότερης διάστασης (*κώδικας*).
2. Κώδικας: Τα συμπιεσμένα δεδομένα.
3. Αποκωδικοποιητής: Ανακατασκευάζει την είσοδο από τον κώδικα.

Η υλοποίηση ενός αυτόματου κωδικοποιητή απαιτεί: έναν μηχανισμό κωδικοποίησης, έναν μηχανισμό αποκωδικοποίησης και μία συνάρτηση απώλειας (loss function), που συγκρίνει την έξοδο με την επιθυμητή τιμή-στόχο. Οι αυτόματοι κωδικοποιητές χρησιμοποιούνται ευρέως στην ανάλυση εικόνας και στην εξαγωγή γενικών πληροφοριών από δεδομένα υψηλής διάστασης.



Σχήμα 2.3: Απλοποιημένο Παράδειγμα Αυτόματου Κωδικοποιητή (Autoencoder) [8]

Ωστόσο, η μη επιβλεπόμενη μάθηση αντιμετωπίζει διάφορες προκλήσεις. Η απουσία ετικετών για την καθοδήγηση της διαδικασίας καθιστά δύσκολη την ερμηνεία και την αξιολόγηση των μοντέλων, ενώ τα αποτελέσματα συχνά εξαρτώνται σε μεγάλο βαθμό από την ποιότητα των δεδομένων. Συνεπώς, μπορεί να είναι απαραίτητη η ανάπτυξη νέων μεθόδων ή τεχνικών για τη βέλτιστη αξιοποίηση των διαθέσιμων δεδομένων.

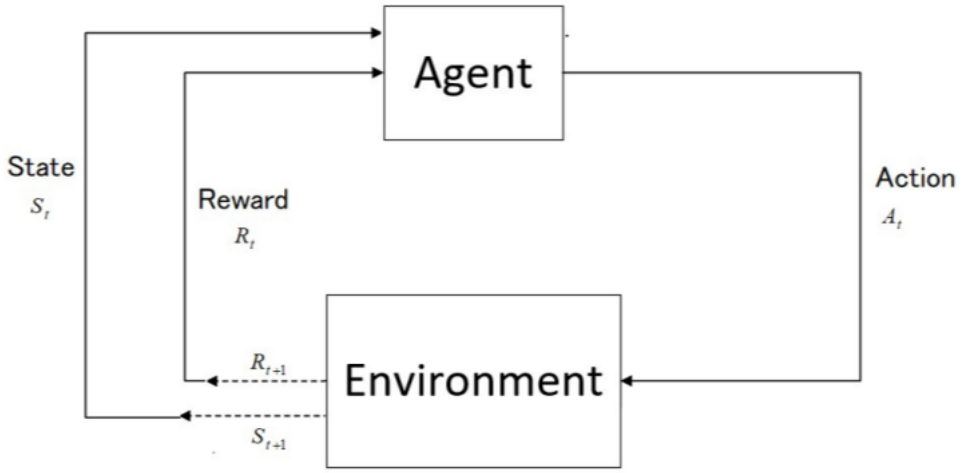
Παρόλα αυτά, η μη επιβλεπόμενη μάθηση έχει πλήθος εφαρμογών σε διάφορους τομείς. Από την ανίχνευση ανωμαλιών έως την εύρεση κρυφών προτύπων σε ιατρικές εικόνες ή βιολογικά δεδομένα για τη διάγνωση και την ανάλυση τάσεων υγείας στην υγειονομική περίθαλψη. Παραδείγματα εφαρμογών στη μηχανική μάθηση περιλαμβάνουν την ανακάλυψη κρυφών δομών σε δεδομένα χρηστών και τη δημιουργία συστάσεων προϊόντων και υπηρεσιών.

2.2.3 Ενισχυτική Μάθηση

Η Ενισχυτική Μάθηση (Reinforcement Learning - RL) [9] αποτελεί έναν από τους πιο συναρπαστικούς και ταχέως εξελισσόμενους τομείς της μηχανικής μάθησης. Πιο συγκεκριμένα, είναι μια κατηγορία μηχανικής μάθησης που επικεντρώνεται στην εκπαίδευση ενός πράκτορα (agent) ώστε να λαμβάνει αποφάσεις που μεγιστοποιούν τη συνολική ανταμοιβή του σε ένα δυναμικό και αβέβαιο περιβάλλον. Δηλαδή, ο πράκτορας αλληλεπιδρά με το περιβάλλον και μαθαίνει μέσω μιας διαδικασίας ανατροφοδότησης, όπου κάθε ενέργεια που εκτελεί επιστρέφει μια ανταμοιβή ή ποινή, προσδιορίζοντας την απόδοσή του. Σε αντίθεση με άλλες μορφές μάθησης, η Ενισχυτική Μάθηση δεν απαιτεί δεδομένα με ετικέτες, αλλά βασίζεται στις συνέπειες των ενεργειών του πράκτορα στο περιβάλλον.

Συνοπτικά, η βασική ιδέα της Ενισχυτικής Μάθησης είναι η βελτιστοποίηση μιας πολιτικής (policy) που καθορίζει τις ενέργειες του πράκτορα, προκειμένου να μεγιστοποιήσει τη συνολική ανταμοιβή που μπορεί να αποκομίσει από το περιβάλλον. Πιο αναλυτικά, σε κάθε κατάσταση του περιβάλλοντος, ο πράκτορας επιλέγει μια ενέργεια, και το περιβάλλον αντιδρά επιστρέφοντας μια ανταμοιβή και μεταβαίνοντας σε μια νέα κατάσταση. Αυτή η διαδικασία βοηθά τον πράκτορα να μάθει ποια πολιτική οδηγεί στη βέλτιστη απόδοση. Η μεγαλύτερη πρόκληση στην Ενισχυτική Μάθηση έγκειται στο γεγονός ότι ο πράκτορας δεν έχει καμία γνώση για

το περιβάλλον· συνεπώς, η στρατηγική του μπορεί να βελτιωθεί μόνο μέσω πειραματισμού και εξερεύνησης, με βάση την ανατροφοδότηση που λαμβάνει.



Σχήμα 2.4: Ανασκόπηση της Ενισχυτικής Μάθησης [9]

Μια πολιτική που ακολουθεί ένας πράκτορας μπορεί να είναι είτε ντετερμινιστική είτε στοχαστική. Στις στοχαστικές πολιτικές, η πολιτική αναπαριστά μια πιθανότητα επιλογής ενεργειών σε κάθε κατάσταση, με στόχο την καλύτερη εξερεύνηση του περιβάλλοντος και την απόκτηση περισσότερης πληροφορίας για τη μελλοντική απόδοση. Αυτό σημαίνει ότι ο πράκτορας πρέπει να αποφασίσει αν θα εκμεταλλευτεί γνωστές καλές ενέργειες ή θα εξερευνήσει νέες, οι οποίες ενδέχεται να αποφέρουν καλύτερα αποτελέσματα στο μέλλον. Οι στοχαστικές πολιτικές επιτρέπουν ευελιξία στη λήψη αποφάσεων: ο πράκτορας επιλέγει από διάφορες ενέργειες με βάση προκαθορισμένες πιθανότητες. Με αυτόν τον τρόπο, ο πράκτορας μπορεί να εξερευνήσει εναλλακτικές στρατηγικές ή να διαχειριστεί καλύτερα απρόβλεπτες καταστάσεις.

Ανάμεσα στους πιο δημοφιλείς αλγόριθμους Ενισχυτικής Μάθησης, ο αλγόριθμος Q-learning προσεγγίζει τη μάθηση μιας πολιτικής μέσω ενός πίνακα Q-τιμών (Q-values). Κάθε ζεύγος κατάστασης-ενέργειας συσχετίζεται με μια εκτίμηση της ανταμοιβής που μπορεί να επιτευχθεί μέσω της εκτέλεσης αυτής της ενέργειας σε αυτή την κατάσταση. Ο αλγόριθμος ενημερώνει και βελτιώνει συνεχώς αυτές τις εκτιμήσεις μέσω επαναλαμβανόμενων αλληλεπιδράσεων με το περιβάλλον. Μοντέλα βασισμένα στη βαθιά μάθηση, γνωστά ως DQN, συνδυάζουν την Ενισχυτική Μάθηση με βαθιά νευρωνικά δίκτυα, διευρύνοντας τις δυνατότητες των πρακτόρων να λειτουργούν σε περιβάλλοντα υψηλών διαστάσεων, όπως η αναγνώριση εικόνων ή η επεξεργασία φυσικής γλώσσας.

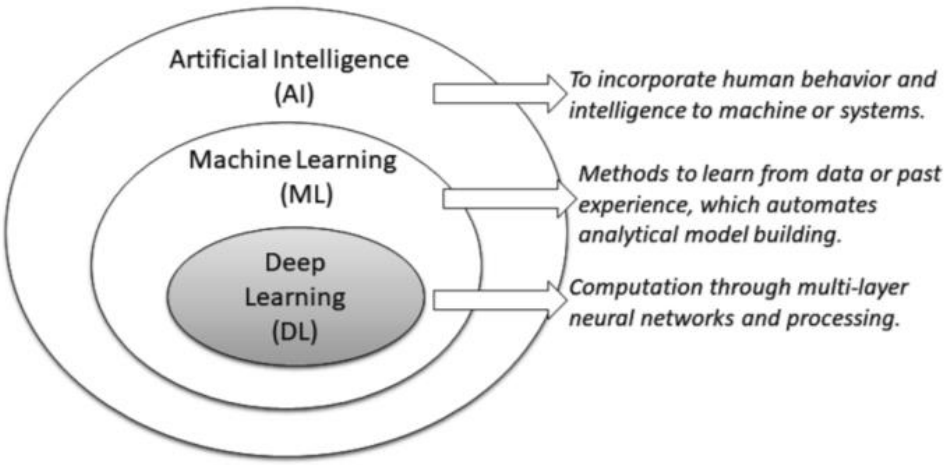
Μία από τις πιο διαδεδομένες και αποτελεσματικές εφαρμογές της Ενισχυτικής Μάθησης στην πραγματική ζωή περιλαμβάνει αυτόνομους πράκτορες, όπως τα αυτοκινούμενα οχήματα και τα ρομπότ, που πρέπει να μαθαίνουν να λαμβάνουν

αποφάσεις σε πραγματικό χρόνο μέσω συνεχούς αλληλεπίδρασης με το περιβάλλον τους. Για παράδειγμα, στα αυτοκινούμενα οχήματα, οι αλγόριθμοι Ενισχυτικής Μάθησης επιτρέπουν στα οχήματα να μαθαίνουν πλοήγηση, αποφυγή εμποδίων και αναγνώριση σημάτων κυκλοφορίας, χωρίς να χρειάζονται προγραμματισμένες οδηγίες. Επιπλέον, η Ενισχυτική Μάθηση έχει ευρεία εφαρμογή στις στρατηγικές παιχνιδιών. Ένα από τα πιο γνωστά παραδείγματα είναι το AlphaGo, το οποίο, μέσω της Ενισχυτικής Μάθησης, νίκησε ανθρώπινους παίκτες, επιδεικνύοντας τη δύναμη της Ενισχυτικής Μάθησης σε σύνθετες καταστάσεις.

Παρότι οι επιτυχίες είναι σημαντικές, η Ενισχυτική Μάθηση παρουσιάζει αρκετές προκλήσεις. Μία από αυτές είναι η ανάγκη για πολλές αλληλεπιδράσεις με το περιβάλλον, γεγονός που καθιστά τη μάθηση υπολογιστικά δαπανηρή, ειδικά σε σύνθετα περιβάλλοντα. Επιπλέον, οι αλγόριθμοι Ενισχυτικής Μάθησης απαιτούν προσεκτική ρύθμιση παραμέτρων και σωστή ισορροπία μεταξύ εξερεύνησης και εκμετάλλευσης για την επίτευξη βέλτιστης απόδοσης. Τέλος, η πολυπλοκότητα που προκύπτει από τις μη γραμμικότητες των δεδομένων πραγματικού κόσμου παραμένει μια από τις μεγαλύτερες προκλήσεις για την ενίσχυση της αξιοπιστίας και της αποδοτικότητας αυτών των συστημάτων.

2.3 Βαθιά Μάθηση

Η Βαθιά Μάθηση [10] αναφέρεται σε έναν από τους πιο σύγχρονους και ισχυρούς τομείς της Τεχνητής Νοημοσύνης. Ως υποπεδίο της μηχανικής μάθησης, η βαθιά μάθηση χρησιμοποιεί τεχνητά νευρωνικά δίκτυα με ελάχιστες ή πολλές κρυφές στρώσεις για τη μοντελοποίηση σύνθετων προτύπων ή σχέσεων μεταξύ των δεδομένων. Μάλιστα, για πολλούς θεωρείται η βασική τεχνολογία της Τέταρτης Βιομηχανικής Επανάστασης.



Σχήμα 2.5: Μια απεικόνιση της θέσης της Βαθιάς Μάθησης (Deep Learning) σε σύγκριση με τη Μηχανική Μάθηση (Machine Learning) και την Τεχνητή Νοημοσύνη (Artificial Intelligence). [10]

Η βασική αρχή της βαθιάς μάθησης είναι η χρήση τεχνητών νευρωνικών δικτύων για τη δημιουργία συστημάτων που μαθαίνουν ιεραρχικές αναπαραστάσεις των δεδομένων. Αυτά τα δίκτυα αποτελούνται από στρώσεις διασυνδεδεμένων νευρώνων, οργανωμένων σε επίπεδα. Κάθε στρώση επεξεργάζεται και εξάγει δεδομένα με πιο αφηρημένα ή υψηλού επιπέδου χαρακτηριστικά. Συγκεκριμένα, οι νευρώνες κάθε στρώσης συνδέονται μέσω βαρών, τα οποία προσαρμόζονται κατά την εκπαίδευση, ώστε να βελτιστοποιείται η λειτουργικότητα του δικτύου.

Ίσως η πιο δημοφιλής αρχιτεκτονική που χρησιμοποιείται στα νευρωνικά δίκτυα της βαθιάς μάθησης είναι τα Συνελικτικά Νευρωνικά Δίκτυα (CNNs). Τα CNNs είναι εξαιρετικά αποτελεσματικά στην αναγνώριση εικόνων, καθώς εκμεταλλεύονται τη χωρική ιεραρχία των δεδομένων μέσω εξελιγμένων τεχνικών. Αυτό επιτρέπει τη μείωση της πολυπλοκότητας του μοντέλου, διατηρώντας παράλληλα κρίσιμα χαρακτηριστικά των δεδομένων. Επιπλέον, τα CNNs αντιμετωπίζουν το πρόβλημα της υπερεκπαίδευσης (overfitting) με τη μέθοδο "dropout." Εφαρμόζονται ευρέως σε ανίχνευση αντικειμένων, αναγνώριση προσώπων και ανάλυση ιατρικών εικόνων, όπου η είσοδος αποτελείται κυρίως από εικόνες. Παραλλαγές όπως τα VGG, AlexNet, Xception, Inception και ResNet καλύπτουν διαφορετικούς τομείς εφαρμογών ανάλογα με τις δυνατότητες μάθησής τους.

Μια άλλη σημαντική κατηγορία νευρωνικών δικτύων είναι τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (RNNs), τα οποία είναι ιδανικά για επεξεργασία δεδομένων χρονικών σειρών ή ακολουθιακών δεδομένων. Χάρη στη "μνήμη" τους, που τους επιτρέπει να ενσωματώνουν προηγούμενες εισόδους ώστε να επηρεάζουν την τρέχουσα είσοδο και έξοδο, τα RNNs μαθαίνουν πρότυπα από τα δεδομένα εκπαίδευσης. Επιπλέον, τα RNNs χρησιμοποιούνται ευρέως στην ανάλυση χρονικών εξαρτήσεων και σε εφαρμογές επεξεργασίας φυσικής γλώσσας, όπως η μηχανική μετάφραση. Οι βελτιωμένες εκδόσεις τους, όπως τα Long Short-Term Memory (LSTM) και Gated Recurrent Units (GRU), βελτιώνουν την απόδοση στις μακροπρόθεσμες εξαρτήσεις, αντιμετωπίζοντας προβλήματα όπως η υποβάθμιση του σήματος.

Ο Πολυεπίπεδος Perceptron (MLP) είναι ένα νευρωνικό δίκτυο εμπρόσθιας ροής (feedforward ANN). Η εκπαίδευση του MLP βασίζεται στον αλγόριθμο Οπισθοδιάδοσης (Backpropagation), ο οποίος θεωρείται βασικό στοιχείο της αρχιτεκτονικής του, μαζί με τις συναρτήσεις ενεργοποίησης όπως οι Tanh, Sigmoid, Softmax και ReLU. Ο MLP αποδίδει εξαιρετικά σε γραμμικά προβλήματα, ενώ κατά την εκπαίδευση χρησιμοποιεί τεχνικές βελτιστοποίησης όπως οι Adam, L-BFGS και SGD.

Μια άλλη σημαντική κατηγορία μοντέλων Βαθιάς Μάθησης είναι τα Γενεσιουργά Ανταγωνιστικά Δίκτυα (GANs), που εισήγαγε ο Ian Goodfellow. Ο στόχος τους είναι η δημιουργία νέων, ρεαλιστικών δειγμάτων κατά παραγγελία. Τα GANs βασίζονται στην αυτόματη αναγνώριση και εκμάθηση μοτίβων ή

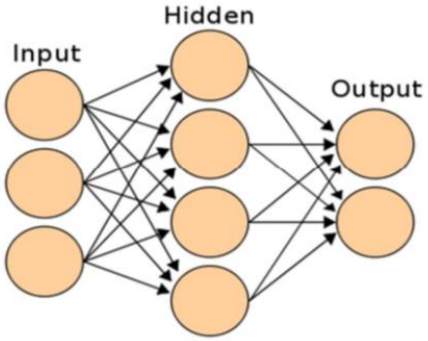
κανονικοτήτων στα δεδομένα εισόδου για τη δημιουργία νέων δειγμάτων από το αρχικό σύνολο δεδομένων. Με την πάροδο του χρόνου, τα GANs εξελίχθηκαν σε έναν γενικό τομέα αυτόνομης δημιουργίας και επέκτασης δεδομένων, παρέχοντας λύσεις σε προβλήματα που απαιτούν γενετικές τεχνικές. Ωστόσο, το θέμα αυτό δεν θα εξεταστεί περαιτέρω στο πλαίσιο αυτής της πτυχιακής.

Παρά τις επιτυχίες της, η Βαθιά Μάθηση αντιμετωπίζει και προκλήσεις. Καταρχάς, η προσβασιμότητά της παραμένει περιορισμένη λόγω της ανάγκης για μεγαλύτερα σύνολα δεδομένων και ισχυρότερους υπολογιστικούς πόρους. Επιπλέον, τα νευρωνικά δίκτυα συχνά λειτουργούν ως "μαύρα κουτιά," γεγονός που εγείρει ηθικά ζητήματα. Η έρευνα στον τομέα της Εξηγήσιμης Τεχνητής Νοημοσύνης (Explainable AI) επιδιώκει να προσφέρει καλύτερη κατανόηση και διαφάνεια στις αποφάσεις που λαμβάνονται από τους αλγορίθμους Βαθιάς Μάθησης.

2.4 Νευρωνικά Δίκτυα

Τα Νευρωνικά Δίκτυα [11] έχουν αναδειχθεί ως ένας από τους πλέον κρίσιμους και ταχύτερα αναπτυσσόμενους τομείς της τεχνητής νοημοσύνης, με εφαρμογές σε διάφορους τομείς όπως η αναγνώριση προτύπων, η κατηγοριοποίηση και η πρόβλεψη. Μάλιστα, είναι εμπνευσμένα από τη θεωρία και τη λειτουργία των στρωμάτων συνδεδεμένων νευρώνων στον ανθρώπινο εγκέφαλο. Συγκεκριμένα, κάθε νευρώνας χρησιμοποιεί συνδέσεις με βάρη, τα οποία προσαρμόζονται κατά τη διάρκεια της εκπαίδευσης, με σκοπό τη μείωση της απόκλισης μεταξύ της προβλεπόμενης και της πραγματικής τιμής.

Αρχικά, περιέχουν αρκετά στρώματα υπολογιστικών κόμβων, γνωστών ως τεχνητοί νευρώνες, οι οποίοι συνεργάζονται για την επίλυση σύνθετων προβλημάτων. Συνήθως αποτελούνται από το στρώμα εισόδου, τα κρυφά στρώματα και το στρώμα εξόδου. Το στρώμα εισόδου δέχεται τα αρχικά δεδομένα και τα στέλνει στα επόμενα στρώματα, όπου τα κρυφά στρώματα επεξεργάζονται τις πληροφορίες χρησιμοποιώντας ReLU, Sigmoid ή Tanh για την ενεργοποίηση. Αυτές οι συναρτήσεις εισάγουν μη-γραμμικότητα στο δίκτυο, επιτρέποντας έτσι την αναγνώριση πολύπλοκων σχέσεων κρυμμένων στα δεδομένα.



Σχήμα 2.6: Απλή περίπτωση αρχιτεκτονικής νευρωνικού δικτύου [11]

Αυτά τα νευρωνικά δίκτυα υπάρχουν σε διάφορες μορφές, κάθε μία με διαφορετική χρήση. Για παράδειγμα, τα απλούστερα νευρωνικά δίκτυα, τα feedforward, στα οποία η ροή των δεδομένων γίνεται μόνο σε μία κατεύθυνση, χρησιμοποιούνται συχνά στην αναγνώριση εικόνας. Παρόμοια, τα Radial Basis Function Networks χρησιμοποιούνται στην αποκατάσταση ενέργειας για την κατηγοριοποίηση δεδομένων με βάση τις αποστάσεις. Επιπλέον, τα Kohonen Networks είναι χρήσιμα στις ιατρικές αναλύσεις λόγω της αυτόνομης φύσης τους για την κατηγοριοποίηση δεδομένων. Τα Convolutional Networks χρησιμοποιούνται κυρίως στην επεξεργασία εικόνας, ενώ τα Recurrent Networks και οι παραλλαγές τους, όπως τα LSTM, επιλύουν προβλήματα χρονικής ακολουθίας. Τέλος, τα Modular Networks ενισχύουν την απόδοση της επεξεργασίας, αποσυνθέτοντας σύνθετες διεργασίες.

Ο κύριος τρόπος υπολογισμού των κλαδών που απαιτούνται για την ενημέρωση των βαρών είναι μέσω οπισθοδρόμησης (backpropagation). Ουσιαστικά, αυτή η ανάστροφη συνεπάγεται ότι τα λάθη που υπολογίζονται στην έξοδο διαχέονται σε όλα τα επίπεδα για την ενημέρωση των βαρών, τα οποία είναι απαραίτητα για την εκπαίδευση βαθιών νευρωνικών δικτύων και καθοριστικά για την ικανότητα τους να προβλέπουν. Αυτή η γενίκευση για πολυεπίπεδα feedforward δίκτυα περιλαμβάνει τον υπολογισμό κλαδών για κάθε επίπεδο μέσω του κανόνα της αλυσίδας. Συμπερασματικά, η οπισθοδρόμηση αποτελεί βασικό συστατικό της συνεχιζόμενης έρευνας στη βελτιστοποίηση νευρωνικών δικτύων και είναι σημαντικός παράγοντας του αλγορίθμου Gauss–Newton.

Ενώ τα βάρη αντικατοπτρίζουν τη δύναμη της σύνδεσης μεταξύ των νευρώνων, τα δεδομένα εκπαίδευσης παρέχουν την απαραίτητη έξοδο για τους μηχανισμούς εκμάθησης. Επειδή τα βάρη τροποποιούνται μέσω της μεθόδου οπισθοδρόμησης, το δίκτυο είναι σε θέση να ελαχιστοποιήσει τα σφάλματα και να επιτύχει μέγιστη ακρίβεια. Πρόκειται για μια κυκλική διαδικασία που το μοντέλο συνεχίζει να βελτιώνει τις προβλέψεις του, ενισχύοντας τις διασυνδεδεμένες οδούς για καλύτερη αποδοτικότητα στη διαχείριση σύνθετων δεδομένων. Επιπλέον, η αρχικοποίηση Xavier και η μέθοδος Gradient Descent είναι κάποιες από τις μεθόδους που βοηθούν στη βελτιστοποίηση της οπισθοδρόμησης, καθιστώντας την εκπαίδευση αρχιτεκτονικών βαθιάς μάθησης πιο σταθερή και ακριβής.

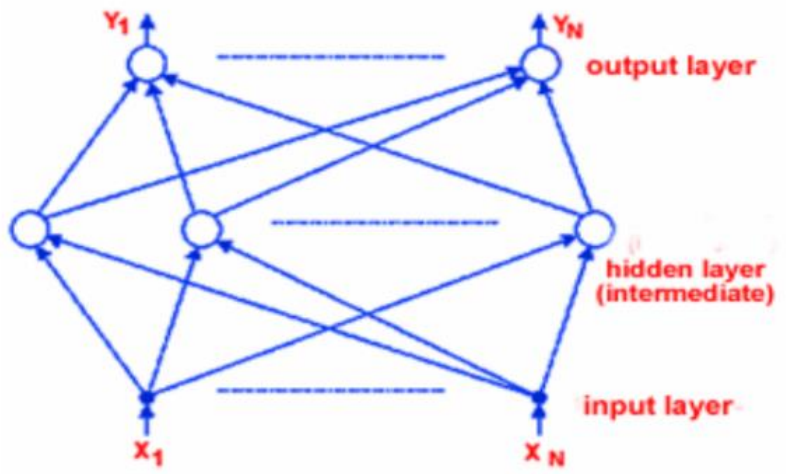
2.4.1 Multilayer Perceptron

Μεταξύ των διαφορετικών παραλλαγών τεχνητών νευρωνικών δικτύων, ένα από τα πιο δημοφιλή είναι ο Multilayer Perceptron (MLP) [12]. Αρχικά, θεωρείται feedforward νευρωνικό δίκτυο, καθώς λειτουργεί μεταφέροντας την πληροφορία από το επίπεδο εισόδου προς το επίπεδο εξόδου, χωρίς να υπάρχει καμία ανατροφοδότηση ή επιστροφή δεδομένων. Ουσιαστικά, βοηθά το δίκτυο να μαθαίνει από τα δεδομένα εισόδου μέσω αλλαγών στις παραμέτρους του κατά τη

διάρκεια της διαδικασίας εκπαίδευσης και ενισχύει την ικανότητα του να κατανοεί σύνθετες και μη γραμμικές σχέσεις μεταξύ διαφορετικών συνόλων δεδομένων.

Ένα απλό MLP αποτελείται από τρία κύρια επίπεδα: το επίπεδο εισόδου, τα κρυμμένα επίπεδα και το επίπεδο εξόδου. Κάθε νευρώνας στο επίπεδο εισόδου αντιστοιχεί σε ένα συγκεκριμένο χαρακτηριστικό του συνόλου δεδομένων εισόδου. Αυτά τα κρυμμένα επίπεδα είναι ενδιάμεσα στάδια που διαδραματίζουν πολύ σημαντικό ρόλο στην επεξεργασία και αποκρυπτογράφηση των σχέσεων που είναι ενσωματωμένες στα δεδομένα. Οι νευρώνες στα κρυμμένα επίπεδα λειτουργούν συνδυαστικά, ώστε το δίκτυο να μπορεί να μοντελοποιήσει πολύπλοκες μη γραμμικές αλληλεπιδράσεις. Τέλος, το επίπεδο εξόδου παρέχει την απόφαση ή την πρόβλεψη, βασισμένη στην εφαρμογή και τον τύπο του προβλήματος που επιλύεται.

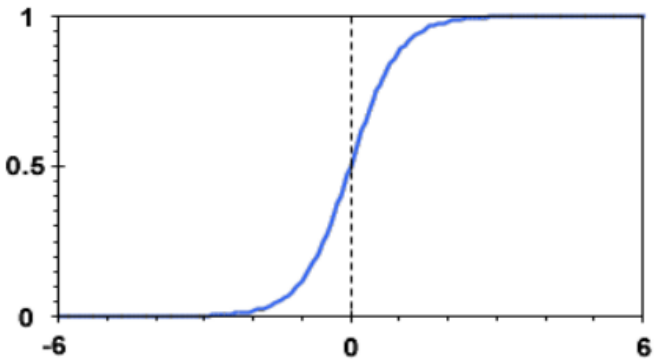
Μία από τις κύριες δυνατότητες του MLP είναι οι σχέσεις μεταξύ των νευρώνων, οι οποίες μετρούνται μέσω των βαρών (weights). Αυτά τα βάρη καθορίζουν την ένταση και την κατεύθυνση της αλληλεπίδρασης μεταξύ των νευρώνων ενός επιπέδου αλλά και των υπολοίπων, επηρεάζοντας τη ροή της πληροφορίας εντός του δικτύου. Ακόμα, αυτά τα βάρη προσαρμόζονται στη διαδικασία εκπαίδευσης μέσω μεθόδων βελτιστοποίησης, όπως ο Gradient Descent, επιτρέποντας στο δίκτυο να βελτιώσει συνολικά την απόδοσή του.



Σχήμα 2.7: Απλή περίπτωση αρχιτεκτονικής Multilayer Perceptron (MLP) [12]

Αφού εκτελέσει έναν γραμμικό μετασχηματισμό της εισόδου, κάθε νευρώνας σε ένα MLP εφαρμόζει μια μη γραμμική συνάρτηση ενεργοποίησης. Παραδείγματα τέτοιων μη γραμμικών συναρτήσεων ενεργοποίησης είναι οι sigmoid, ReLU (Rectified Linear Unit) και tanh, οι οποίες επιτρέπουν στο δίκτυο να μαθαίνει μη γραμμικές σχέσεις, κάτι που διαφορετικά θα ήταν αδύνατο. Δηλαδή, χωρίς αυτές τις μη γραμμικές συναρτήσεις ενεργοποίησης, το MLP θα ήταν περιορισμένο στην επεξεργασία μόνο γραμμικών αλληλεπιδράσεων, γεγονός που θα περιόριζε σοβαρά την ικανότητά του να παρέχει πρακτικές λύσεις σε

προβλήματα που απαιτούν τη μοντελοποίηση πιο σύνθετων μη γραμμικών σχέσεων.



Σχήμα 2.8: Η σιγμοειδής (sigmoid) συνάρτηση ενεργοποίησης [12]

Η εκπαίδευση ενός MLP πραγματοποιείται μέσω της προσαρμογής των βαρών στις συνδέσεις μεταξύ των νευρώνων. Θεωρητικά, η εκπαίδευση βασίζεται στον αλγόριθμο Οπισθοδρόμησης (Backpropagation), ο οποίος εκτιμά το σφάλμα στο επίπεδο εξόδου και το διαδίδει προς τα πίσω μέσα στο δίκτυο, προκειμένου να ενημερωθούν τα βάρη στις συνδέσεις. Μία από τις κύριες μεθόδους για την ενημέρωση αυτών των βαρών είναι ο Gradient Descent, η οποία υπολογίζει την κλίση του σφάλματος ως προς τα βάρη για να προσδιορίσει την κατεύθυνση και το μέγεθος των απαραίτητων προσαρμογών. Μάλιστα, αυτή η διαδικασία επαναλαμβάνεται έως ότου το δίκτυο φτάσει στη βέλτιστη λύση, μειώνοντας έτσι το σφάλμα πρόβλεψης. Ακόμα, συνήθως απαιτούνται μεγάλα σύνολα δεδομένων, ώστε το μοντέλο να μπορεί να εντοπίσει τα κατάλληλα μοτίβα που ταιριάζουν καλύτερα στη συγκεκριμένη περίπτωση. Η μαθησιακή ταχύτητα (learning rate), η ορμή (momentum) και οι αλγόριθμοι βελτιστοποίησης μπορούν να συμβάλουν στη βελτίωση της απόδοσης του αλγορίθμου εκπαίδευσης και να διευκολύνουν την ομαλότερη σύγκλιση.

Λόγω της εκτεταμένης πρακτικής τους χρησιμότητας, οι τομείς εφαρμογής των MLP εκτείνονται από καθαρά επιστημονικούς έως και πολύ πρακτικούς τομείς της καθημερινής ζωής. Αρχικά, στην επιχειρηματική και βιομηχανική ανάλυση, χρησιμοποιούνται για την ανάλυση μεγάλων δεδομένων, την πρόβλεψη τάσεων, καθώς και για πολλές άλλες χρήσεις. Ύστερα, στην ιατρική μπορούν να αξιοποιηθούν για τη διάγνωση ασθενειών μέσω της ανάλυσης ιατρικών εικόνων. Επιπλέον, στην επεξεργασία φυσικής γλώσσας, διευκολύνουν τη δημιουργία συστημάτων αναγνώρισης φωνής και μετάφρασης. Ακόμα, στην όραση υπολογιστών, αξιοποιούνται για την αναγνώριση προσώπων ή αντικειμένων, παρέχοντας έτσι λύσεις σε διάφορα καθημερινά και επιστημονικά προβλήματα.

Συμπερασματικά, οι εφαρμογές τους έχουν οδηγήσει σε πολλές νέες επιτεύξεις σε διάφορους τομείς και έχουν βελτιώσει την ικανότητα ορισμένων υπολογιστικών συστημάτων να διαχειρίζονται σύνθετα προβλήματα με εξαιρετική

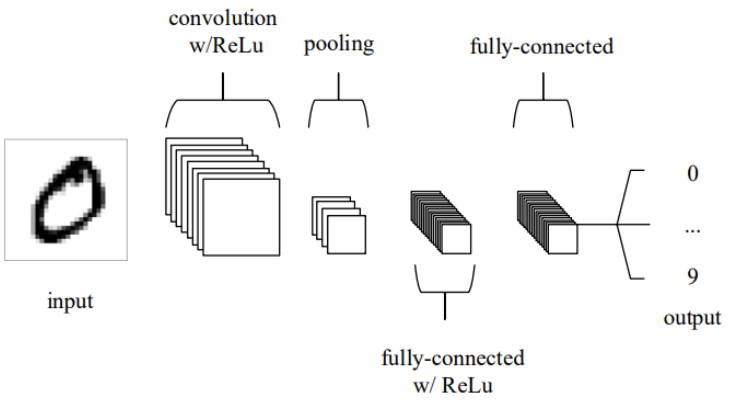
απόδοση. Ωστόσο, χρησιμοποιούνται κυρίως σε συνδυασμό με άλλες τεχνικές και αρχιτεκτονικές, όπως τα Convolutional Neural Networks (CNNs) και τα Recurrent Neural Networks (RNNs), καθώς ορισμένες εφαρμογές επιτυγχάνουν πολύ καλύτερα αποτελέσματα με αυτόν τον τρόπο.

2.4.2 Συνελικτικά Νευρωνικά Δίκτυα

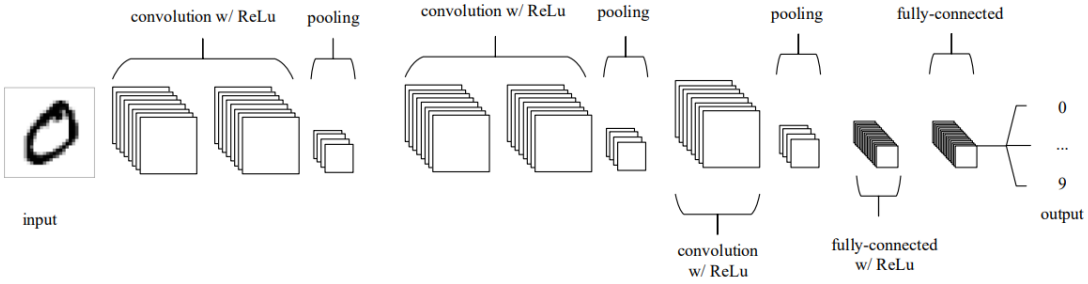
Μια ξεχωριστή αρχιτεκτονική τεχνητών νευρωνικών δικτύων, που έχει σχεδιαστεί ειδικά για την ανάλυση δεδομένων εικόνας, είναι τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs) [13]. Αυτά τα δίκτυα χρησιμοποιούν συνελικτικές λειτουργίες για την ανίχνευση χαρακτηριστικών μέσα στις εικόνες, διευκολύνοντας όχι μόνο τη γρήγορη ανάλυση αλλά και την ταξινόμηση σύνθετων μοτίβων. Επιπλέον, η θεμελιώδης αρχή πίσω από τα CNNs ενισχύει τη δυνατότητα γενίκευσης των αποτελεσμάτων, ενώ ταυτόχρονα μειώνει τον αριθμό των απαιτούμενων παραμέτρων.

Αρχικά, η δομή των συνελικτικών δικτύων αποτελείται από τρία βασικά στρώματα: τα συνελικτικά στρώματα, τα στρώματα υποδειγματοληψίας (pooling) και τα πλήρως συνδεδεμένα στρώματα. Στα συνελικτικά στρώματα, διαφόρων ειδών φίλτρα ή πυρήνες εφαρμόζονται στις εισερχόμενες εικόνες για την εξαγωγή τοπικών χαρακτηριστικών, όπως οι άκρες, οι γωνίες και διάφορα μοτίβα. Επιπλέον, τα στρώματα υποδειγματοληψίας μειώνουν τις χωρικές διαστάσεις των δεδομένων, ενισχύοντας την αποδοτικότητα και περιορίζοντας τον κίνδυνο υπερεκπαίδευσης. Τέλος, τα πλήρως συνδεδεμένα στρώματα αξιοποιούν τις εξαγόμενες πληροφορίες για την τελική ταξινόμηση ή πρόβλεψη, ενδυναμώνοντας τη γενίκευση των αποτελεσμάτων.

Οι χάρτες ενεργοποίησης, γνωστοί και ως συναρτήσεις ενεργοποίησης, αναδεικνύουν τα χαρακτηριστικά που θεωρεί σημαντικά το μοντέλο για μια συγκεκριμένη ταξινόμηση. Με τη διατήρηση της μη γραμμικότητας της εισόδου, συναρτήσεις ενεργοποίησης όπως η ReLU (Rectified Linear Unit) επιτρέπουν στα συνελικτικά δίκτυα να αναλύουν πιο σύνθετα μοτίβα.



Σχήμα 2.9: Απλή αρχιτεκτονική Συνελικτικού Νευρωνικού Δικτύου (CNN) με 5 στρώματα [13]



Σχήμα 2.10: Περίπλοκη αρχιτεκτονική Συνελικτικού Νευρωνικού Δικτύου (CNN) [13]

Ένα βασικό πλεονέκτημα των συνελικτικών δικτύων είναι η ικανότητα τους να επαναχρησιμοποιούν παραμέτρους μέσω της τεχνικής διαμοιρασμού παραμέτρων. Αυτή η προσέγγιση επιτρέπει σε κάθε φίλτρο να ανταποκρίνεται σε παρόμοια χαρακτηριστικά σε διαφορετικές περιοχές της εικόνας, επιταχύνοντας τη διαδικασία εκπαίδευσης και μειώνοντας σημαντικά την υπολογιστική πολυπλοκότητα του μοντέλου.

Η υπερεκπαίδευση αποτελεί σημαντική πρόκληση στα συνελικτικά νευρωνικά δίκτυα. Ένα μοντέλο που προσαρμόζεται υπερβολικά στο σύνολο εκπαίδευσης μπορεί να αποτύχει να γενικεύσει αποτελεσματικά σε μη ορατά δεδομένα. Για την αντιμετώπιση αυτού του ζητήματος, μπορούν να εφαρμοστούν στρατηγικές όπως το dropout, η κανονικοποίηση και η αύξηση δεδομένων.

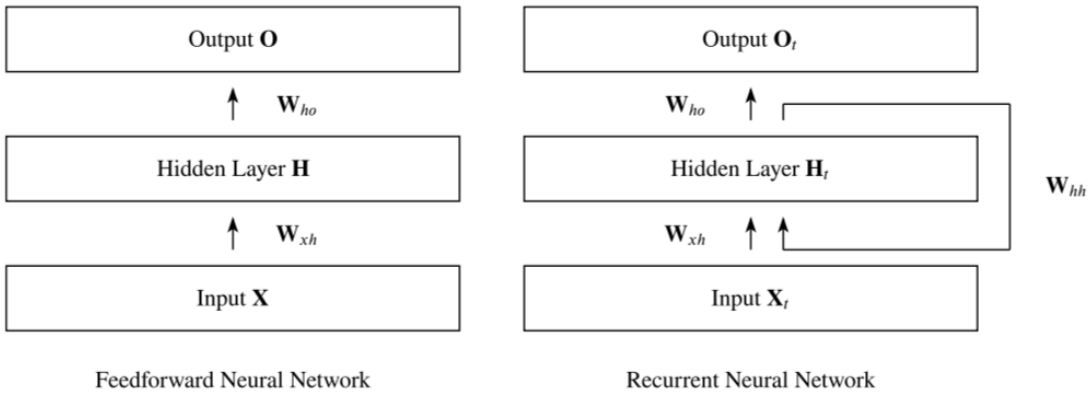
Επιπλέον, αν και τα συνελικτικά δίκτυα είναι εξαιρετικά αποτελεσματικά για την ανάλυση εικόνων, συνεπάγονται υψηλό υπολογιστικό κόστος. Για τη διαχείριση μεγάλων εικόνων, όπως αυτών στο σύνολο δεδομένων ImageNet, είναι απαραίτητες τεχνικές όπως η μείωση ανάλυσης και η κατανομή φίλτρων σε πολλαπλά επίπεδα.

2.4.3 Επαναλαμβανόμενα Νευρωνικά Δίκτυα

Μια εξαιρετικά χρήσιμη κατηγορία νευρωνικών δικτύων που έχει σχεδιαστεί για την ανάλυση ακολουθιακών δεδομένων είναι τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNNs) [14]. Τα RNNs μπορούν να θεωρηθούν ότι διαθέτουν μια δυναμική «κατάσταση μνήμης» που ανανεώνεται συνεχώς, επιτρέποντάς τους να διατηρούν πληροφορίες από προηγούμενες εισόδους. Αυτή η ιδιότητα καθιστά τα RNNs ιδιαίτερα κατάλληλα για εφαρμογές όπως η πρόβλεψη χρονοσειρών, η ανάλυση κειμένου και η αναγνώριση φωνής, οι οποίες απαιτούν την εξέταση χρονικών εξαρτήσεων.

Ένα θεμελιώδες χαρακτηριστικό των RNNs, που διευκολύνει τη μεταφορά πληροφοριών μεταξύ χρονικών βημάτων, είναι η ύπαρξη αναδρομικών συνδέσεων μεταξύ των νευρώνων. Παρ' όλα αυτά, ο ίδιος αυτός μηχανισμός

μπορεί να συμβάλλει και στις προαναφερθείσες προκλήσεις. Συγκεκριμένα, όταν μια σειρά συναρτήσεων ενεργοποίησης και πολλαπλασιασμών μητρών οδηγεί στη μείωση ή την υπερβολική αύξηση των παραγώγων, επηρεάζεται αρνητικά η απόδοση του δικτύου, προκαλώντας τα γνωστά προβλήματα της εξαφάνισης ή της έκρηξης παραγώγων κατά τη διαδικασία εκπαίδευσης.

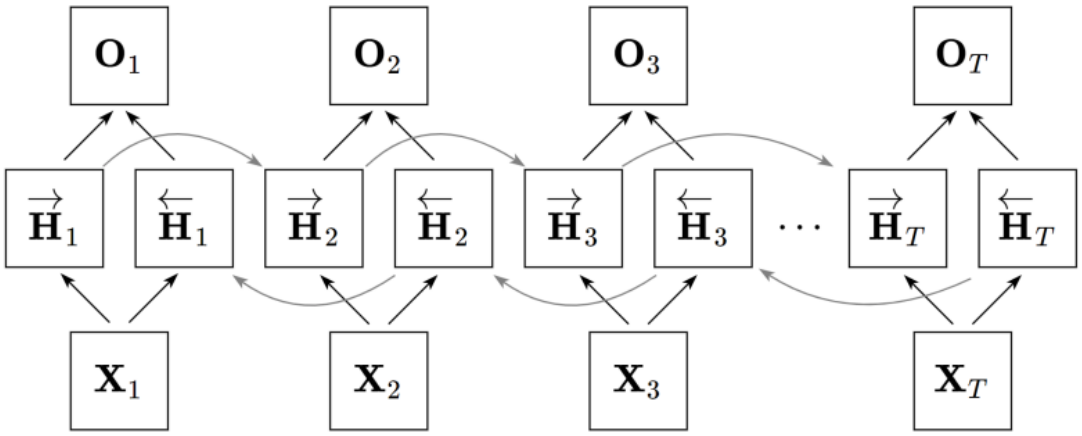


Σχήμα 2.11: Οπτικοποίηση των διαφορών μεταξύ Feedforward Νευρωνικού Δικτύου και RNN [14]

Για να καταστεί δυνατή η παραδοσιακή οπισθοδιάδοση, η τεχνική που είναι γνωστή ως «Οπισθοδιάδοση Μέσω Χρόνου» (Backpropagation Through Time - BPTT) περιλαμβάνει την αποσύνθεση του επαναλαμβανόμενου νευρωνικού δικτύου σε κάθε χρονικό βήμα. Ωστόσο, λόγω των προαναφερθέντων προκλήσεων που σχετίζονται με τις παραγώγους, χρησιμοποιείται συχνά μια παραλλαγή, η «Περιορισμένη Οπισθοδιάδοση Μέσω Χρόνου» (Truncated BPTT), η οποία περιορίζει την οπισθοδιάδοση σε έναν καθορισμένο αριθμό χρονικών βημάτων.

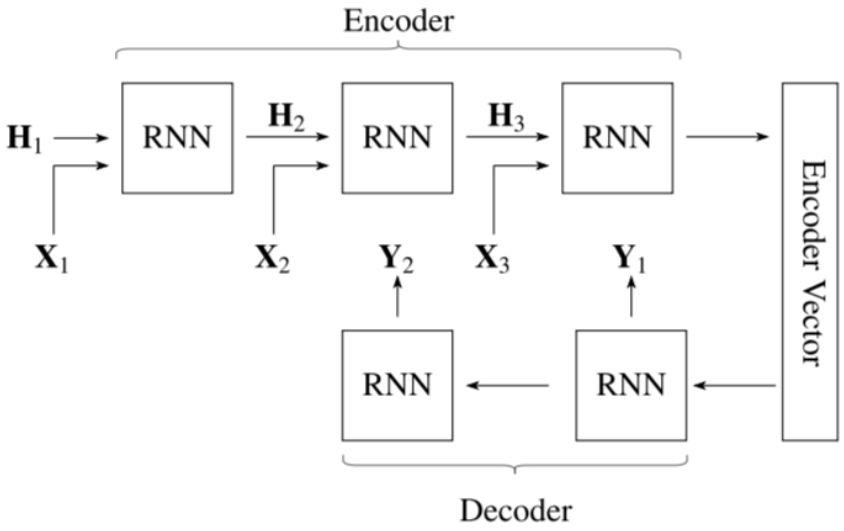
Μία από τις σημαντικές καινοτομίες στα RNNs είναι η εισαγωγή των Μονάδων Μακράς και Βραχείας Μνήμης (Long Short-Term Memory, LSTM). Οι LSTM χρησιμοποιούν πύλες εισόδου, εξόδου και λήθης για την αποτελεσματική διαχείριση της ροής των πληροφοριών. Ο σχεδιασμός αυτός επιλύει το πρόβλημα της εξαφάνισης των παραγώγων, επιτρέποντας τη διατήρηση πληροφοριών για εκτεταμένα χρονικά διαστήματα. Ως εκ τούτου, τα RNNs έχουν σημειώσει σημαντική επιτυχία στις εφαρμογές LSTM για σύνθετα καθήκοντα, όπως η ανάλυση συναισθημάτων και η αυτόματη μετάφραση.

Τα Διπλής Κατεύθυνσης Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Bidirectional RNNs) εισάγουν μια καινοτόμα διάσταση στη μοντελοποίηση ακολουθιακών δεδομένων, επιτρέποντας τη ροή της πληροφορίας τόσο προς τα εμπρός όσο και προς τα πίσω. Αυτή η διπλή κατεύθυνση αξιοποιεί το περιβαλλοντικό δεδομένο τόσο από προηγούμενα όσο και από επόμενα στοιχεία εντός ενός δεδομένου χρονικού πλαισίου. Αυτή η προσέγγιση είναι ιδιαίτερα επωφελής για καθήκοντα που απαιτούν συμπλήρωση προτάσεων και αναγνώριση προτύπων.



Σχήμα 2.12: Αρχιτεκτονική Διπλής Κατεύθυνσης Επαναλαμβανόμενου Νευρωνικού Δικτύου (Bidirectional RNN) [14]

Επιπλέον, οι εξελίξεις όπως τα δίκτυα κωδικοποιητή-αποκωδικοποιητή και οι αρχιτεκτονικές που ενσωματώνουν Μηχανισμούς Προσοχής (Attention Mechanisms) αποτελούν σημαντική πρόοδο στο σχεδιασμό των RNN. Αυτές οι καινοτομίες έχουν επεκτείνει την εφαρμογή των RNN σε πιο σύνθετους τομείς, όπως η παραγωγή κειμένου, η επεξήγηση εικόνων και η μηχανική μετάφραση. Η ενσωμάτωση του μηχανισμού προσοχής διευκολύνει την προτεραιότητα συγκεκριμένων τμημάτων εισόδου, οδηγώντας σε σημαντικές βελτιώσεις στην ακρίβεια και την αποδοτικότητα.



Σχήμα 2.13: Οπτικοποίηση του μοντέλου Sequence to Sequence (seq2seq) με RNNs [14]

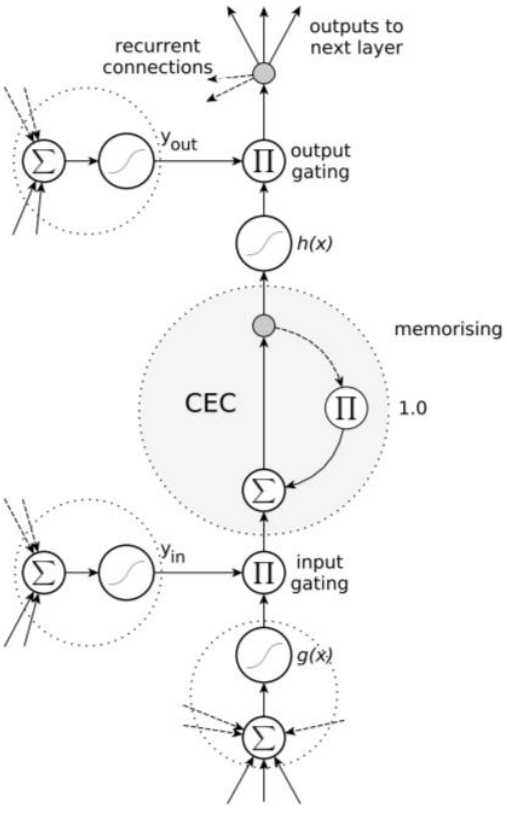
2.4.4 Long Short-Term Memory Νευρωνικά Δίκτυα

Ένα εξειδικευμένο παρακλάδι των επαναλαμβανόμενων νευρωνικών δικτύων (RNNs), γνωστό ως δίκτυα Μακράς και Βραχείας Διάρκειας Μνήμης (Long Short-Term Memory - LSTMs) [15], έχει αναπτυχθεί για να αντιμετωπίσει τις προκλήσεις της εξαφάνισης και της έκρηξης των παραγώγων κατά τη διαδικασία εκπαίδευσης.

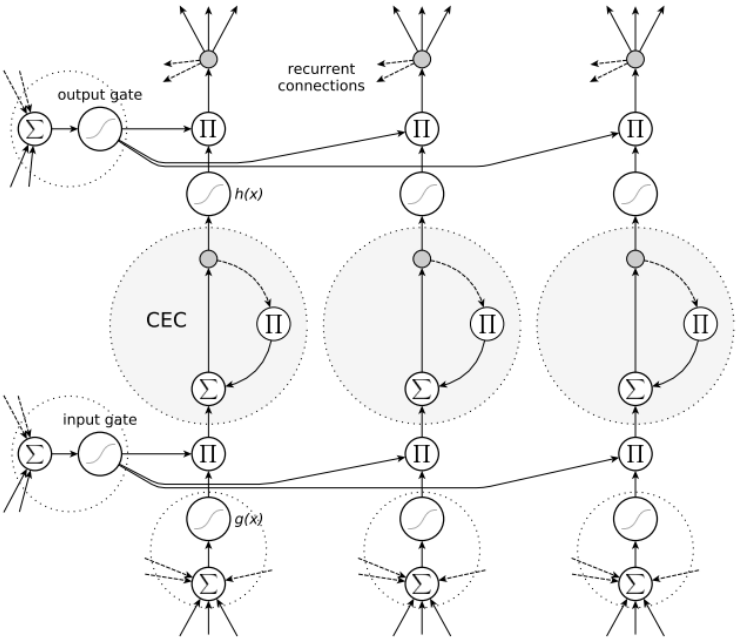
Τα LSTMs διευκολύνουν τη μακροπρόθεσμη διατήρηση σχετικών πληροφοριών μέσω της αρχιτεκτονικής τους, που περιλαμβάνει κύτταρα μνήμης και πύλες που διαχειρίζονται τη ροή της πληροφορίας. Κατά συνέπεια, τα LSTMs είναι ιδιαίτερα αποτελεσματικά σε εφαρμογές όπως η μηχανική μετάφραση, η αναγνώριση ομιλίας και η πρόβλεψη σειρών χρόνου.

Το βασικό στοιχείο ενός LSTM είναι το μπλοκ μνήμης, το οποίο περιλαμβάνει ένα κύτταρο μνήμης μαζί με τρεις κύριες πύλες: την πύλη εισόδου, την πύλη εξόδου και την πύλη λήθης. Η πύλη εισόδου είναι υπεύθυνη για την επιλογή των νέων πληροφοριών που θα αποθηκευτούν στο κύτταρο μνήμης, ενώ η πύλη εξόδου ρυθμίζει την πληροφορία που εξέρχεται από το κύτταρο. Για την πρόληψη της συσσώρευσης μη σχετικών δεδομένων, η πύλη λήθης επιτρέπει την απομάκρυνση ξεπερασμένων ή άσχετων πληροφοριών.

Για τη διασφάλιση της διατήρησης δεδομένων μεγάλης διάρκειας, ενσωματώνεται ο Καθολικός Κύκλος Σφαλμάτων (Constant Error Carousel, CEC), επιτρέποντας την αδιάλειπτη ροή σφαλμάτων μέσα στο κύτταρο μνήμης. Τα Δίκτυα Μακράς και Βραχείας Διάρκειας Μνήμης (LSTM) είναι ιδιαίτερα κατάλληλα για τη διαχείριση δεδομένων με μακροχρόνιες εξαρτήσεις λόγω της αρχιτεκτονικής τους, που υπερβαίνει τις περιορισμένες χρονικές ικανότητες των παραδοσιακών επαναλαμβανόμενων νευρωνικών δικτύων (RNNs). Η δομή CEC ενισχύεται περαιτέρω από πολλαπλασιαστικές πύλες, οι οποίες ρυθμίζουν τον χρόνο και τον τρόπο μεταφοράς σημάτων.



Σχήμα 2.14: Τυπικό μπλοκ μνήμης LSTM [15]



Σχήμα 2.15: Μπλοκ μνήμης LSTM με τρία κύτταρα και επαναλαμβανόμενες αυτοσυνδέσεις [15]

Σε μια υβριδική προσέγγιση εκπαίδευσης, τα LSTMs χρησιμοποιούν εκμάθηση σε πραγματικό χρόνο εντός των κυττάρων μνήμης, ενώ εφαρμόζουν οπισθοδιάδοση μέσω χρόνου για την έξοδο. Αυτή η μεθοδολογία εξασφαλίζει αποτελεσματική εκπαίδευση του δικτύου, διατηρώντας την προβλεπτική ακρίβεια, ακόμη και όταν τα δεδομένα εισόδου συλλέγονται σε παρατεταμένα και συνεχόμενα διαστήματα.

Με την πάροδο του χρόνου, αυτή η οικογένεια δικτύων έχει διαφοροποιηθεί σε διάφορες αρχιτεκτονικές που επεκτείνουν τη βασική τους σχεδίαση με πολλούς τρόπους, όπως τα Grid LSTMs και τα Διπλής Κατεύθυνσης LSTMs. Τα Διπλής Κατεύθυνσης LSTMs επιτρέπουν την ανάλυση και στις δύο κατευθύνσεις, βελτιώνοντας την κατανόηση του πλαισίου, ενώ τα Grid LSTMs διευκολύνουν την επικοινωνία μεταξύ πολλαπλών διαστάσεων δεδομένων.

2.5 Βελτιστοποιητές Νευρωνικών Δικτύων

Η επιλογή του Βελτιστοποιητή [16] σε ένα νευρωνικό δίκτυο παίζει καθοριστικό ρόλο στην αποτελεσματικότητα της εκπαίδευσης και στην τελική απόδοση του μοντέλου. Συγκεκριμένα, οι βελτιστοποιητές βασίζονται κυρίως σε πρώτης τάξης μεθόδους που αξιοποιούν την πληροφορία των παραγώγων της συνάρτησης απώλειας. Ο στόχος είναι η εύρεση ενός ελαχίστου της συνάρτησης απώλειας, συχνά μέσω μιας ακολουθίας ενημερώσεων των παραμέτρων του μοντέλου. Οι πιο συνηθισμένοι βελτιστοποιητές διαφέρουν κυρίως ως προς τον τρόπο ενημέρωσης των παραμέτρων και τη διαχείριση της πληροφορίας από προηγούμενα βήματα.

Ο Stochastic Gradient Descent (SGD) είναι ο πιο βασικός βελτιστοποιητής, που βασίζεται στην απλή ενημέρωση των παραμέτρων προς την κατεύθυνση της αντίθετης παραγώγου. Αν και απλός, έχει ορισμένα μειονεκτήματα, όπως η μεγάλη ευαισθησία στην επιλογή του ρυθμού μάθησης. Για να αντιμετωπιστεί αυτό το πρόβλημα, η μέθοδος Momentum εισάγει έναν όρο ορμής που επιτρέπει στη βελτιστοποίηση να διατηρεί μια κατεύθυνση κίνησης, μειώνοντας έτσι τις διακυμάνσεις και επιταχύνοντας τη σύγκλιση σε βαθιές κοιλάδες.

Η προσαρμοστική προσέγγιση του RMSProp επιλύει το πρόβλημα της διακύμανσης του ρυθμού μάθησης προσαρμόζοντας τον ρυθμό μάθησης για κάθε παράμετρο ξεχωριστά. Ενώ ο SGD χρησιμοποιεί έναν σταθερό ρυθμό, το RMSProp λαμβάνει υπόψη τις ιστορικές πληροφορίες της κλίσης, βελτιώνοντας τη σύγκλιση σε περίπλοκες συναρτήσεις απώλειας. Μια παραλλαγή αυτής της μεθόδου είναι το Adam, που συνδυάζει το Momentum και το RMSProp, εξασφαλίζοντας μια ισορροπία μεταξύ σταθερότητας και ταχύτητας μάθησης.

Ο Adam είναι ένας από τους πιο διαδεδομένους βελτιστοποιητές, καθώς χρησιμοποιεί προσαρμοστικούς ρυθμούς μάθησης και παραμέτρους ορμής. Ειδικότερα, αξιοποιεί εκθετικά κινητούς μέσους όρους τόσο της πρώτης όσο και της δεύτερης ροπής της κλίσης. Ωστόσο, διάφορες έρευνες έχουν δείξει ότι ο Adam μπορεί να μην είναι πάντα η καλύτερη επιλογή, καθώς σε ορισμένα προβλήματα υπολείπεται έναντι άλλων μεθόδων, όπως το Momentum ή το SGD με σωστά ρυθμισμένα υπερπαραμέτρους.

Ο Nadam είναι μια βελτιωμένη εκδοχή του Adam, ενσωματώνοντας την επιτάχυνση του Nesterov. Αυτή η τεχνική επιτρέπει στο μοντέλο να έχει καλύτερη προσέγγιση των βέλτιστων σημείων και να συγκλίνει ταχύτερα. Παρόλο που θεωρείται ανώτερος από τον Adam σε ορισμένες περιπτώσεις, η πραγματική του απόδοση εξαρτάται έντονα από την κατάλληλη ρύθμιση των υπερπαραμέτρων του.

Η επιλογή του κατάλληλου βελτιστοποιητή δεν είναι απλή υπόθεση και εξαρτάται από διάφορους παράγοντες, όπως η φύση του προβλήματος, η διαθεσιμότητα υπολογιστικών πόρων και η ανάγκη για γενίκευση. Η επίδραση των υπερπαραμέτρων μπορεί να είναι καθοριστική, με τον ρυθμό μάθησης να αποτελεί τον πιο σημαντικό παράγοντα. Στη βιβλιογραφία, υπάρχουν περιπτώσεις όπου μικρές αλλαγές σε παραμέτρους όπως το β_1 και το β_2 του Adam μπορούν να αλλάξουν ριζικά τα αποτελέσματα.

Η υπερπαραμετροποίηση των βελτιστοποιητών είναι μια ιδιαίτερα κρίσιμη διαδικασία. Σε αρκετές μελέτες, παρατηρείται ότι η επίδοση ενός βελτιστοποιητή μπορεί να αλλάξει δραματικά όταν ρυθμιστούν κατάλληλα παράμετροι όπως το ε του Adam ή ο ρυθμός μείωσης του ρυθμού μάθησης. Αυτή η παρατήρηση καταδεικνύει ότι καμία μέθοδος δεν είναι παγκοσμίως ανώτερη, αλλά εξαρτάται από τη σωστή επιλογή υπερπαραμέτρων.

Στην πράξη, η σύγκριση των βελτιστοποιητών μπορεί να είναι παραπλανητική αν δεν λαμβάνονται υπόψη οι λεπτομέρειες της ρύθμισης τους. Έρευνες έχουν δείξει ότι η τάση να συγκρίνονται βελτιστοποιητές με τις προεπιλεγμένες τιμές των υπερπαραμέτρων τους οδηγεί συχνά σε λανθασμένα συμπεράσματα. Αντίθετα, όταν κάθε βελτιστοποιητής λαμβάνει τη βέλτιστη ρύθμιση, οι διαφορές μεταξύ τους μειώνονται σημαντικά.

Τέλος, η επιλογή του κατάλληλου βελτιστοποιητή εξαρτάται από το εκάστοτε πρόβλημα και το διαθέσιμο υπολογιστικό κόστος. Για μεγάλα νευρωνικά δίκτυα, προσεγγίσεις όπως ο Adam ή το RMSProp μπορεί να είναι πιο αποδοτικές, ενώ για απλούστερα προβλήματα, ο SGD με κατάλληλες ρυθμίσεις μπορεί να δώσει εξίσου καλά ή και καλύτερα αποτελέσματα. Η βελτιστοποίηση των νευρωνικών δικτύων παραμένει ενεργό ερευνητικό πεδίο, με νέες τεχνικές να προτείνονται διαρκώς για τη βελτίωση της απόδοσης και της αποδοτικότητας των υπαρχόντων μεθόδων.

SGD(H_t, η_t)

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell(\theta_t)$$

MOMENTUM(H_t, η_t, γ)

$$v_0 = 0$$
$$v_{t+1} = \gamma v_t + \nabla \ell(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta_t v_{t+1}$$

NESTEROV(H_t, η_t, γ)

$$v_0 = 0$$
$$v_{t+1} = \gamma v_t + \nabla \ell(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta_t \left(\gamma v_{t+1} + \nabla \ell(\theta_t)\right)$$

RMSPROP$(H_t, \eta_t, \gamma, \rho, \epsilon)$

$$v_0 = 1, m_0 = 0$$
$$v_{t+1} = \rho v_t + (1-\rho)\nabla \ell(\theta_t)^2$$
$$m_{t+1} = \gamma m_t + \frac{\eta_t}{\sqrt{v_{t+1} + \epsilon}} \nabla \ell(\theta_t)$$
$$\theta_{t+1} = \theta_t - m_{t+1}$$

ADAM$(H_t, \alpha_t, \beta_1, \beta_2, \epsilon)$

$$m_0 = 0, v_0 = 0$$
$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla \ell(\theta_t)$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla \ell(\theta_t)^2$$
$$b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}}$$
$$\theta_{t+1} = \theta_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon} b_{t+1}$$

NADAM$(H_t, \alpha_t, \beta_1, \beta_2, \epsilon)$

$$m_0 = 0, v_0 = 0$$
$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla \ell(\theta_t)$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla \ell(\theta_t)^2$$
$$b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}}$$
$$\theta_{t+1} = \theta_t - \alpha_t \frac{\beta_1 m_{t+1} + (1-\beta_1)\nabla \ell(\theta_t)}{\sqrt{v_{t+1}} + \epsilon} b_{t+1}$$

Σχήμα 2.16: Σύγκριση αλγορίθμων βελτιστοποίησης (SGD, Momentum, Nesterov, RMSProp, Adam, Nadam) για εκπαίδευση νευρωνικών δικτύων [16]

3 Αριθμητική Ανάλυση και Μέθοδοι Runge-Kutta

3.1 Αριθμητική Ανάλυση

Η Αριθμητική Ανάλυση [17] είναι ένας θεμελιώδης τομέας των μαθηματικών που ασχολείται με τη μελέτη και ανάπτυξη αλγορίθμων για την εκτίμηση και επίλυση μαθηματικών προβλημάτων μέσω υπολογιστικών μεθόδων. Η κύρια πρόκληση της αριθμητικής ανάλυσης είναι να προσφέρει προσεγγιστικές λύσεις σε προβλήματα που δεν μπορούν να λυθούν ακριβώς ή που η ακριβής λύση είναι υπολογιστικά αδύνατη, λόγω του μεγάλου όγκου δεδομένων ή της πολυπλοκότητας του συστήματος.

Μεταξύ των διαφόρων τύπων αλγορίθμων που χρησιμοποιούνται στην αριθμητική ανάλυση περιλαμβάνονται: οι μέθοδοι ολοκλήρωσης, ο εντοπισμός ριζών και οι λύσεις συστημάτων γραμμικών εξισώσεων. Μάλιστα, ένας από τους πρώτους τρόπους προσέγγισης τετραγωνικών ριζών, που αποτελεί παραλλαγή της μεθόδου του Νεύτωνα, είναι ο τύπος του Ηρών. Αποτελεί χαρακτηριστικό παράδειγμα σταθερής επαναληπτικής διαδικασίας, η οποία διακρίνεται για την ακρίβεια και τη σταθερότητα της.

Ένα ακόμη κρίσιμο ζήτημα στους αλγορίθμους, που συνήθως καθορίζει την αξιοπιστία τους, είναι η σταθερότητα. Ένας ασταθής αλγόριθμος μπορεί να παράγει μη ρεαλιστικά αποτελέσματα, ακόμη και αν η αρχική προσέγγιση βρίσκεται κοντά στη λύση. Η σταθερότητα διασφαλίζεται συχνά μέσω τεχνικών όπως η κλιμάκωση ή η επιλογή βελτιστοποιημένων αρχικών τιμών, οι οποίες συμβάλλουν επίσης στην αποδοτικότητα.

Ίσως η πιο διαδεδομένη μέθοδος για την επίλυση γραμμικών συστημάτων είναι η απαλοιφή Gauss, όπου η διαδικασία περιλαμβάνει τη μείωση ενός συστήματος σε τριγωνική μορφή και την επίλυσή του με αναδρομική υποκατάσταση. Επιπλέον, τεχνικές παραγοντοποίησης, όπως η LU παραγοντοποίηση, προσφέρουν μεγάλη ευελιξία και αποδοτικότητα, επιτρέποντας την ενσωμάτωσή τους σε πιο σύνθετα υπολογιστικά συστήματα.

Η μέθοδος του Newton αποτελεί τη βάση για έναν μεγάλο αριθμό τεχνικών που χρησιμοποιούνται στην επίλυση μη γραμμικών εξισώσεων. Χαρακτηρίζεται από την ταχεία σύγκλισή της προς τη λύση, υπό την προϋπόθεση ότι η αρχική προσέγγιση βρίσκεται αρκετά κοντά στη σωστή λύση, και αξιοποιεί παραγώγους στους υπολογισμούς της. Η αποδοτικότητα αυτής της μεθόδου επηρεάζεται τόσο από την αρχική προσέγγιση όσο και από τα χαρακτηριστικά της εμπλεκόμενης συνάρτησης, επιτρέποντάς της να αποδίδει εξαιρετικά σε ποικίλα πλαίσια.

Η αριθμητική ολοκλήρωση περιλαμβάνει τεχνικές όπως ο κανόνας του τραπεζίου και η μέθοδος του Simpson, που χρησιμοποιούνται για την εκτίμηση του εμβαδού κάτω από καμπύλες. Αυτές οι μέθοδοι λειτουργούν βασικά με

παρόμοιο τρόπο: διαχωρίζουν το διάστημα ολοκλήρωσης σε μικρότερα τμήματα και χρησιμοποιούν πολυωνυμικές προσεγγίσεις για να αυξήσουν την ακρίβεια.

Ένας σημαντικός τομέας της αριθμητικής ανάλυσης είναι η επίλυση διαφορικών εξισώσεων. Ανάμεσα στις πιο διαδεδομένες τεχνικές συγκαταλέγονται οι μέθοδοι Runge-Kutta, οι οποίες έχουν σχεδιαστεί για την προσέγγιση λύσεων εξισώσεων οποιασδήποτε τάξης. Αυτές οι επαναληπτικές μέθοδοι διακρίνονται για την υψηλή ακρίβειά τους, ακόμη και όταν εφαρμόζονται σε συστήματα εξισώσεων, και δεν απαιτούν τη χρήση παραγώγων υψηλότερης τάξης, γεγονός που τις καθιστά ιδιαίτερα πρακτικές.

Η μέθοδος Runge-Kutta τέταρτης τάξης χαίρει ευρείας αναγνώρισης για την αποτελεσματική ισορροπία που προσφέρει μεταξύ ακρίβειας και υπολογιστικής αποδοτικότητας. Αυτή η μέθοδος χρησιμοποιεί τέσσερα ενδιάμεσα βήματα για να επιτύχει υψηλό επίπεδο ακρίβειας στην προσέγγιση λύσεων. Η ευελιξία της την καθιστά ιδιαίτερα κατάλληλη για διάφορες εφαρμογές στις φυσικές επιστήμες, την τεχνολογία, αλλά και σε αναδυόμενους τομείς όπως η μηχανική μάθηση.

Τα προβλήματα ιδιοτιμών έχουν σημαντική σημασία, ιδιαίτερα στην ανάλυση δυναμικών συστημάτων. Τεχνικές όπως το θεώρημα των κύκλων του Gershgorin και η μείωση σε μορφή Hessenberg διευκολύνουν αποδοτικούς και ακριβείς υπολογισμούς, ενισχύοντας έτσι την ικανότητα επίλυσης συστημάτων υψηλών διαστάσεων.

Επιπλέον, οι τεχνικές διακριτοποίησης επιτρέπουν την αναπαράσταση προβλημάτων σε χώρους πεπερασμένων διαστάσεων, κάτι που είναι ζωτικής σημασίας για την επίλυση μερικών διαφορικών εξισώσεων μέσω μεθόδων φυσικής προσομοίωσης, όπως οι πεπερασμένες διαφορές και τα πεπερασμένα στοιχεία.

Η θεωρία σφάλματος αποτελεί ένα από τα θεμελιώδη εργαλεία για την κατανόηση της συμπεριφοράς των αριθμητικών αλγορίθμων. Για τη βελτιστοποίηση των αλγορίθμων και την αξιολόγηση της αξιοπιστίας των αποτελεσμάτων τους, οι μαθηματικοί οφείλουν να αναλύουν τα σφάλματα στρογγυλοποίησης και να υπολογίζουν όρια για το συνολικό σφάλμα.

Συνεπώς, η αριθμητική ανάλυση παρέχει τα απαραίτητα εργαλεία στους υπολογιστές για την επίλυση σύνθετων μαθηματικών προβλημάτων. Επιπλέον, έχει διαδραματίσει σημαντικό ρόλο στη συνεχή ανάπτυξη νέων τεχνικών και τη βελτίωση των υφιστάμενων, ώστε να ανταποκριθεί στις σύγχρονες επιστημονικές και τεχνολογικές προκλήσεις.

3.2 Διαφορικές Εξισώσεις

Ένα θεμελιώδες εργαλείο στη μαθηματική ανάλυση είναι η διατύπωση σχέσεων μέσω μιας μόνο μεταβλητής και των παραγώγων της, που συνήθως

εκφράζονται μέσω Διαφορικών Εξισώσεων [18]. Γενικά, γίνεται διάκριση μεταξύ απλών και μερικών διαφορικών εξισώσεων, επειδή οι πρώτες περιγράφουν εξαρτήσεις από μία ανεξάρτητη μεταβλητή και τις ανώτερες παραγώγους της, ενώ οι δεύτερες αφορούν παραγώγους που εξαρτώνται από πολλαπλές μεταβλητές. Αυτή η διάκριση είναι κρίσιμη για τη χρήση τους ως προς την μοντελοποίηση ποικίλων φαινομένων σε πεδία όπως η φυσική, η βιολογία και η οικονομία.

Οι διαφορικές εξισώσεις ταξινομούνται βάσει της μορφής και της δομής τους. Για παράδειγμα, στις γραμμικές διαφορικές εξισώσεις, οι εξαρτημένες μεταβλητές και οι παράγωγοί τους εκφράζονται με απλό γραμμικό τρόπο, ενώ στις μη γραμμικές εξισώσεις, αυτές οι μεταβλητές συνδέονται μέσω πιο σύνθετων σχέσεων. Η τάξη μιας διαφορικής εξίσωσης καθορίζεται από την υψηλότερη παράγωγο που περιέχει, ενώ ο βαθμός ορίζεται από τον εκθέτη αυτής της υψηλότερης παραγώγου.

Η χρήση των διαφορικών εξισώσεων είναι διαδεδομένη σε πλήθος επιστημονικών κλάδων. Στον τομέα της φυσικής, αυτές οι εξισώσεις είναι απαραίτητες για τη μοντελοποίηση δυναμικών συστημάτων, όπως η κίνηση των σωματιδίων σύμφωνα με τη μηχανική του Νεύτωνα. Στη μηχανική, τα θεωρητικά πλαίσια που παρέχουν οι εξισώσεις του Euler και του Hamilton διευκολύνουν την ανάλυση σύνθετων φαινομένων. Στις βιολογικές επιστήμες, οι εξισώσεις δυναμικής πληθυσμών, όπως τα μοντέλα Lotka-Volterra, εξετάζουν τις αλληλεπιδράσεις μεταξύ διαφορετικών ειδών.

Οι λύσεις των διαφορικών εξισώσεων διακρίνονται σε δύο κύριες κατηγορίες: αναλυτικές και αριθμητικές. Οι αναλυτικές λύσεις, αν και ακριβείς, περιορίζονται συνήθως σε απλούστερα σενάρια. Αντίθετα, οι αριθμητικές μέθοδοι παρέχουν αποτελεσματικές προσεγγίσεις για πιο σύνθετα προβλήματα, με τη μέθοδο Runge-Kutta να αποτελεί χαρακτηριστικό παράδειγμα.

Η επίδραση των διαφορικών εξισώσεων στην τεχνολογία είναι σημαντική. Στον τομέα της ηλεκτροδυναμικής, οι εξισώσεις του Maxwell περιγράφουν τη συμπεριφορά των ηλεκτρομαγνητικών πεδίων, ενώ στη χρηματοοικονομική, η εξίσωση Black-Scholes χρησιμοποιείται για την τιμολόγηση παραγώγων. Επιπλέον, στην κβαντομηχανική, η εξίσωση Schrödinger παρέχει κρίσιμες γνώσεις για τη δυναμική των σωματιδίων.

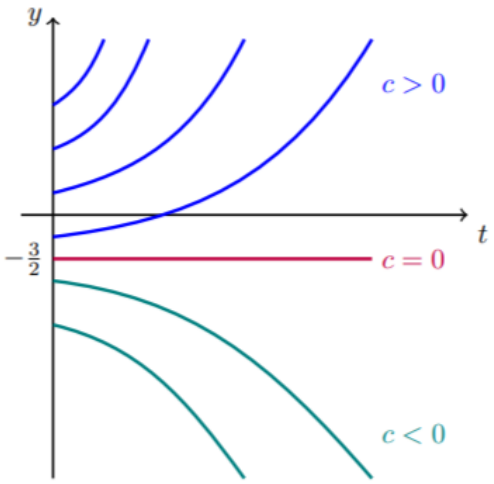
Η ευελιξία των διαφορικών εξισώσεων στην αντιμετώπιση πραγματικών προβλημάτων έχει οδηγήσει στη μεγάλη δημοτικότητα τους. Η βασική τους ιδέα αποτελεί θεμέλιο για την ανάλυση σύνθετων φαινομένων, από τον σχεδιασμό γεφυρών έως την κατανόηση βιολογικών συστημάτων. Αν και οι περισσότερες εξισώσεις δεν μπορούν να λυθούν απευθείας, οι προσεγγιστικές μέθοδοι επιτρέπουν την κατανόηση και την πρακτική εφαρμογή των λύσεων τους.

3.2.1 Συνήθεις Διαφορικές Εξισώσεις

Στη φυσική, τη μηχανική και τις υπολογιστικές επιστήμες, οι Συνήθεις Διαφορικές Εξισώσεις (ΣΔΕ) [19] αποτελούν θεμελιώδες εργαλείο μαθηματικής ανάλυσης και χρησιμοποιούνται εκτενώς στη διαμόρφωση και επίλυση μοντέλων. Αυτές οι εξισώσεις περιγράφουν μια άγνωστη συνάρτηση που εξαρτάται από μία ανεξάρτητη μεταβλητή και τις παραγώγους της. Συγκεκριμένα η τάξη μιας ΣΔΕ καθορίζεται από την υψηλότερη παράγωγο που εμφανίζεται σε αυτήν. Για παράδειγμα, μια διαφορική εξίσωση πρώτης τάξης της μορφής: $y' = f(x, y)$ περιγράφει τη χρονική εξέλιξη ενός συστήματος, ενώ μια γενική εξίσωση δεύτερης τάξης: $y'' + a(x)y' + b(x)y = f(x)$, εμφανίζεται σε φυσικά φαινόμενα όπως η διάδοση κυμάτων και οι μηχανικές ταλαντώσεις.

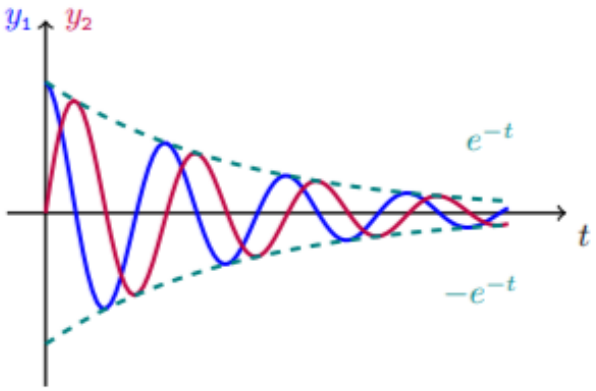
Αρχικά, είναι εξαιρετικά σημαντικό να κατανοηθεί η διαφορά μεταξύ γραμμικών και μη γραμμικών ΣΔΕ [20]. Μια εξίσωση χαρακτηρίζεται ως γραμμική όταν η εξαρτημένη μεταβλητή και οι παράγωγοι της εμφανίζονται σε γραμμική μορφή. Για παράδειγμα, η εξίσωση: $y' + 2y = e^x$ είναι μια γραμμική ΣΔΕ πρώτης τάξης, ενώ η εξίσωση: $y' = y^2 + x$ είναι μη γραμμική λόγω του τετραγωνικού όρου. Μάλιστα, οι γραμμικές εξισώσεις υπακούουν στην αρχή της επαλληλίας, επιτρέποντας τη σύνθεση γενικών λύσεων από ήδη υπάρχουσες μέσω γραμμικών συνδυασμών. Αντίθετα, η ανάλυση μη γραμμικών εξισώσεων είναι πολύ πιο περίπλοκη, καθώς αυτές συχνά εμφανίζουν φαινόμενα όπως πολλαπλά σημεία ισορροπίας και μη αναλυτικές λύσεις.

Για την επίλυση ΣΔΕ πρώτης τάξης, έχουν αναπτυχθεί διάφορες μέθοδοι. Για παράδειγμα οι διαχωρίσιμες εξισώσεις της μορφής: $\frac{dy}{dx} = g(x)h(y)$ μπορούν να επιλυθούν μέσω ολοκλήρωσης, αφού οι μεταβλητές μπορούν να διαχωριστούν. Επίσης, για τις ακριβείς εξισώσεις της μορφής: $P(x, y)dx + Q(x, y)dy = 0$, η ύπαρξη μιας δυναμικής συνάρτησης που ορίζει τις λύσεις εξασφαλίζεται από την προϋπόθεση πως $\frac{dP}{dy} = \frac{dQ}{dx}$. Μάλιστα, όταν το κριτήριο ακρίβειας δεν ικανοποιείται, συχνά αναζητείται ένας ολοκληρωτικός παράγοντας που μετατρέπει την εξίσωση σε ακριβή μορφή. Συγκεκριμένα, η εξίσωση πολλαπλασιάζεται με μια κατάλληλα επιλεγμένη συνάρτηση $\mu(x)$ μετατρέποντας τη σε ολοκληρώσιμη μορφή. Ακόμα, η εξίσωση Bernoulli, που δίνεται στη γενική μορφή: $y' + p(x)y = q(x)y^r$, επιλύεται μέσω της αντικατάστασης: $z = y^{1-r}$, η οποία τη μετατρέπει (για $r = 0$) σε γραμμική εξίσωση ή (για $r = 1$) σε χωριζομένων μεταβλητών.



Σχήμα 3.1: Μερικές λύσεις της γραμμικής ΣΔΕ: $y' = 2y + 3$ για διαφορετικές τιμές του c, όπου η λύση είναι η $y(t) = ce^{2t} - \frac{3}{2}, c \in R$ [19]

Για την επίλυση ΣΔΕ δεύτερης τάξης, η γραμμική ομογενής εξίσωση με σταθερούς συντελεστές της μορφής: $y'' + ay' + by = 0$ εξαρτάται από τις ρίζες του χαρακτηριστικού πολυωνύμου $\lambda^2 + a\lambda + b\lambda = 0$. Αν οι ρίζες είναι διαφορετικές (ελέγχοντας την ανεξαρτησία τους με την ορίζουσα Wronski) και πραγματικές, η λύση αποτελείται από δύο εκθετικούς όρους και είναι της μορφής: $y(x) = c1\, e^{\lambda 1 x} + c2\, e^{\lambda 2 x}$. Αντίθετα, αν οι ρίζες είναι μιγαδικές, η λύση παίρνει τη μορφή: $y(x) = c1\, e^{\alpha x} cos(\beta x) + c2\, e^{\alpha x} sin(\beta x)$, δηλαδή περιέχει ημίτονα και συνημίτονα πολλαπλασιασμένα με εκθετική συνάρτηση. Τέλος, αν υπάρχει διπλή ρίζα, η λύση είναι της μορφής: $c1 + c2\, x(e^{\lambda x})$, ώστε να καλύπτει την ιδιαιτερότητα της επαναλαμβανόμενης ρίζας.



Σχήμα 3.2: Γραφική αναπαράσταση λύσεων της ΣΔΕ: $y'' + 2y' + 6y = 0$ για πραγματικές θεμελιώδεις λύσεις: $y_1(t) = e^{-t}\, cos(\sqrt{5}t)$ & $y_2(t) = e^{-t}\, sin(\sqrt{5}t)$ [19]

Μια άλλη σημαντική τεχνική επίλυσης ΣΔΕ είναι ο μετασχηματισμός Laplace. Αυτός επιτρέπει τη μετατροπή μιας διαφορικής εξίσωσης σε μια αλγεβρική

εξίσωση στη μετασχηματισμένη περιοχή, η οποία μπορεί συχνά να λυθεί πιο εύκολα. Η μέθοδος είναι ιδιαίτερα χρήσιμη στην ανάλυση ηλεκτρικών κυκλωμάτων και μηχανικών συστημάτων.

$f(t)$	$F(s) = \mathcal{L}[f(t)]$	D_F
$f(t) = 1$	$F(s) = \frac{1}{s}$	$s > 0$
$f(t) = e^{at}$	$F(s) = \frac{1}{(s-a)}$	$s > a$
$f(t) = t^n$	$F(s) = \frac{n!}{s^{(n+1)}}$	$s > 0$
$f(t) = \sin(at)$	$F(s) = \frac{a}{s^2+a^2}$	$s > 0$
$f(t) = \cos(at)$	$F(s) = \frac{s}{s^2+a^2}$	$s > 0$
$f(t) = \sinh(at)$	$F(s) = \frac{a}{s^2-a^2}$	$s > \lvert a \rvert$
$f(t) = \cosh(at)$	$F(s) = \frac{s}{s^2-a^2}$	$s > \lvert a \rvert$
$f(t) = t^n e^{at}$	$F(s) = \frac{n!}{(s-a)^{(n+1)}}$	$s > a$
$f(t) = e^{at}\sin(bt)$	$F(s) = \frac{b}{(s-a)^2+b^2}$	$s > a$
$f(t) = e^{at}\cos(bt)$	$F(s) = \frac{(s-a)}{(s-a)^2+b^2}$	$s > a$
$f(t) = e^{at}\sinh(bt)$	$F(s) = \frac{b}{(s-a)^2-b^2}$	$s - a > \lvert b \rvert$
$f(t) = e^{at}\cosh(bt)$	$F(s) = \frac{(s-a)}{(s-a)^2-b^2}$	$s - a > \lvert b \rvert$

Σχήμα 3.3: Πίνακας με Μετασχηματισμούς Laplace γνωστών συναρτήσεων [19]

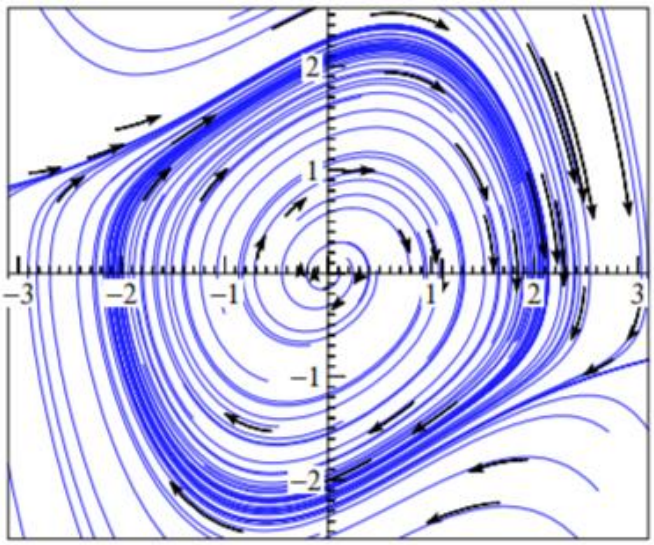
Ουσιαστικά, οι ΣΔΕ είναι καθοριστικές για τη μελέτη της σταθερότητας αυτόνομων συστημάτων, όπου η συμπεριφορά του συστήματος καθορίζεται από ένα σύνολο εξισώσεων. Για παράδειγμα, συστήματα πληθυσμιακής δυναμικής, μηχανικά συστήματα ταλάντωσης, οικονομικά μοντέλα, αλλά και άλλα φυσικά και τεχνολογικά συστήματα, συχνά περιγράφονται μέσω ΣΔΕ. Η ανάλυση της σταθερότητας περιλαμβάνει τη μελέτη των ιδιοτιμών των συστημάτων αυτών, καθώς και τη χρήση διαγραμμάτων φάσης, για να κατανοήσουμε τη συμπεριφορά του συστήματος στο χρόνο. Επιπλέον, αυτή η ανάλυση μας βοηθά να καθορίσουμε αν το σύστημα θα καταλήξει σε μια σταθερή κατάσταση, θα ταλαντώνεται ή θα αποκλίνει, προσφέροντας πολύτιμες πληροφορίες για τη μακροχρόνια δυναμική του.

3.2.2 Προβλήματα Αρχικών Τιμών για Συνήθεις Διαφορικές Εξισώσεις

Τα Προβλήματα Αρχικών Τιμών (ΠΑΤ) [21] αποτελούν θεμελιώδη αντικείμενα μελέτης στις Συνήθεις Διαφορικές Εξισώσεις (ΣΔΕ), καθώς περιγράφουν φυσικά και τεχνητά δυναμικά συστήματα μέσω διαφορικών σχέσεων. Αρχικά, ένα ΠΑΤ ορίζεται από μια εξίσωση της μορφής: $y' = f(x, y)$ με δεδομένη αρχική συνθήκη $y(x_0) = y_0$, όπου η συνάρτηση $y(x)$ περιγράφει την εξέλιξη του συστήματος στη μεταβλητή x. Ουσιαστικά, τέτοια προβλήματα έχουν καθοριστικό ρόλο σε επιστημονικά πεδία όπως η φυσική, η βιολογία και η οικονομία, καθώς μοντελοποιούν καταστάσεις όπου η συμπεριφορά ενός συστήματος εξαρτάται από την προηγούμενη κατάσταση του.

Όσον αφορά την ύπαρξη και τη μοναδικότητα των λύσεων, αυτές εξαρτώνται άμεσα από τις ιδιότητες της συνάρτησης $f(x, y)$. Αν η f είναι συνεχώς διαφορίσιμη και ικανοποιεί τη συνθήκη Lipschitz, τότε εγγυάται μοναδική λύση σύμφωνα με το θεώρημα Picard-Lindelöf. Σε αντίθετη περίπτωση, αν η f είναι μόνο συνεχής, μπορεί να υπάρχουν πολλαπλές λύσεις ή ακόμη και να μην υπάρχει καθόλου λύση. Συνεπώς, η μελέτη της κατηγορίας της συνάρτησης f είναι κρίσιμη για την κατανόηση της συμπεριφοράς του συστήματος και την επιλογή κατάλληλων μεθόδων επίλυσης.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι αυτόνομες διαφορικές εξισώσεις, στις οποίες η συνάρτηση $f(y)$ δεν εξαρτάται ρητά από τη μεταβλητή x. Αυτό επιτρέπει τη γεωμετρική αναπαράσταση της συμπεριφοράς τους μέσω διαγραμμάτων φάσης, διευκολύνοντας τη μελέτη της δυναμικής του συστήματος. Για παράδειγμα, ο ταλαντωτής Van der Pol αποτελεί χαρακτηριστικό μη γραμμικό σύστημα που εμφανίζει ένα σταθερό οριακό κύκλο. Αυτό το φαινόμενο υποδηλώνει ότι ανεξαρτήτως της αρχικής κατάστασης, οι λύσεις τείνουν σε μια περιοδική τροχιά, γεγονός που έχει σημαντικές εφαρμογές στη φυσική και τη μηχανική.



Σχήμα 3.4: Λύσεις στον φασικό χώρο του ταλαντωτή Van der Pol για $\mu = 0.4$ [21]

Αναλύοντας τα γραμμικά συστήματα με σταθερούς συντελεστές, παρατηρούμε ότι η λύση τους εκφράζεται μέσω του εκθετικού πίνακα e^{Ax}, όπου A είναι ο πίνακας των συντελεστών. Αν οι ιδιοτιμές του A έχουν αρνητικό πραγματικό μέρος, τότε η λύση συγκλίνει στο μηδέν, καθιστώντας το σύστημα ασυμπτωτικά σταθερό. Αντιθέτως, αν υπάρχουν ιδιοτιμές με θετικό πραγματικό μέρος, η λύση αποκλίνει εκθετικά, οδηγώντας σε ασταθή συστήματα. Επομένως, η ανάλυση των ιδιοτιμών παρέχει μια χρήσιμη προσέγγιση για την πρόβλεψη της μακροχρόνιας συμπεριφοράς ενός συστήματος.

Για τα μη γραμμικά συστήματα, η σταθερότητα εξετάζεται μέσω της ιδιοτιμής του Ιακωβιανού πίνακα (ή αλλιώς πίνακας Jacobi) της συνάρτησης $f(y)$ στο σημείο ισορροπίας. Συμπερασματικά, αν όλες οι ιδιοτιμές έχουν αρνητικά πραγματικά μέρη, το σύστημα είναι ασυμπτωτικά σταθερό, ενώ αν υπάρχει έστω και μία με θετικό πραγματικό μέρος, το σημείο ισορροπίας είναι ασταθές. Αυτή η μέθοδος επιτρέπει την κατηγοριοποίηση των σταθερών σημείων ενός δυναμικού συστήματος σε κέντρα, ελκυστές και απωθητές, γεγονός που είναι χρήσιμο στην ανάλυση πολύπλοκων δυναμικών φαινομένων.

Η αριθμητική επίλυση των ΠΑΤ είναι κρίσιμη, καθώς οι αναλυτικές λύσεις είναι διαθέσιμες μόνο σε συγκεκριμένες περιπτώσεις. Οι κλασικές μέθοδοι όπως η Euler, η Runge-Kutta και η Adams-Bashforth χρησιμοποιούνται για την προσεγγιστική επίλυση. Ειδικότερα, οι μέθοδοι Runge-Kutta επιτυγχάνουν υψηλότερη ακρίβεια για δεδομένο βήμα ολοκλήρωσης, χωρίς την ανάγκη υπολογισμού παραγώγων υψηλότερης τάξης. Συνεπώς, οι μέθοδοι αυτές αποτελούν μία από τις πιο αποδοτικές επιλογές για αριθμητικές προσεγγίσεις.

Ιδιαίτερη προσοχή απαιτείται στα δύσκαμπτα (stiff) προβλήματα, όπου η ύπαρξη διαφορετικών χρονικών κλιμάκων προκαλεί αριθμητικές δυσκολίες. Για την αποτελεσματική επίλυση τους, χρησιμοποιούνται ασαφείς μέθοδοι, όπως οι πίσω Euler ή οι Rosenbrock, που επιτρέπουν τη λήψη μεγαλύτερων βημάτων χωρίς την αποσταθεροποίηση της λύσης. Αυτό καθιστά τις μεθόδους αυτές εξαιρετικά χρήσιμες στη μελέτη συστημάτων με έντονες μεταβολές στις δυναμικές τους ιδιότητες.

Πέρα από τις συμβατικές διαφορικές εξισώσεις, αξίζει να σημειωθεί ότι τα διαφορικά-αλγεβρικά συστήματα εμφανίζονται σε πολλές εφαρμογές, όπως η μηχανική και ο έλεγχος συστημάτων. Αυτά τα συστήματα περιλαμβάνουν τόσο διαφορικές όσο και αλγεβρικές εξισώσεις, γεγονός που απαιτεί εξειδικευμένες τεχνικές επίλυσης, καθώς δεν είναι πάντα δυνατή η μετατροπή τους σε συστήματα συνήθων διαφορικών εξισώσεων.

Τέλος, ιδιαίτερη σημασία έχουν οι διαφορικές εξισώσεις σε πολλαπλότητες, οι οποίες περιγράφουν τη δυναμική συστημάτων που υπόκεινται σε φυσικούς ή τεχνητούς περιορισμούς. Σημαντικό είναι ότι τέτοιες εξισώσεις απαιτούν την εφαρμογή θεωρίας διαφορίσιμων πολλαπλοτήτων και εξειδικευμένων αριθμητικών μεθόδων που σέβονται τη γεωμετρία του προβλήματος.

Συνοψίζοντας, η κατανόηση και η επίλυση των προβλημάτων αρχικών τιμών είναι κρίσιμες τόσο στη θεωρητική όσο και στην πρακτική εφαρμογή των διαφορικών εξισώσεων.

3.3 Σειρές Taylor

Οι Σειρές Taylor [22] αποτελούν ένα θεμελιώδες μαθηματικό εργαλείο που επιτρέπει την αναπαράσταση συναρτήσεων μέσω άπειρων αθροισμάτων. Με άλλα λόγια, μια πολύπλοκη συνάρτηση μπορεί να εκφραστεί ως μια δυναμοσειρά, δηλαδή ως ένα άθροισμα πολυωνυμικών όρων με συντελεστές που εξαρτώνται από τις παραγώγους της συνάρτησης σε κάποιο σημείο. Επομένως, η ανάπτυξη Taylor δίνει τη δυνατότητα προσεγγιστικού υπολογισμού συναρτήσεων με τη χρήση πεπερασμένων όρων, γεγονός που είναι ιδιαίτερα χρήσιμο στις αριθμητικές μεθόδους και στις επιστημονικές εφαρμογές.

Για να μπορεί μια συνάρτηση να αναπτυχθεί σε σειρά Taylor γύρω από ένα σημείο $x = \alpha$, πρέπει να είναι αναλυτική, δηλαδή να μπορεί να γραφτεί ως:

$$f(x) = \sum_{n=0}^{\infty} C_n (x - \alpha)^n$$

Πιο συγκεκριμένα, οι συντελεστές C_n δίνονται από τη σχέση: $C_n = \frac{f^{(n)}(\alpha)}{n!}$.

Αξίζει να σημειωθεί ότι η σύγκλιση της σειράς εξαρτάται από την ακτίνα σύγκλισης, η οποία καθορίζει για ποιες τιμές του x η σειρά προσεγγίζει σωστά τη συνάρτηση. Επιπλέον, αν και πολλές κοινές συναρτήσεις έχουν απεριόριστη ακτίνα σύγκλισης, υπάρχουν και περιπτώσεις όπου η σειρά Taylor συγκλίνει μόνο εντός ενός συγκεκριμένου διαστήματος.

Ειδικότερα, ένα σημαντικό είδος σειράς Taylor είναι η σειρά Maclaurin, η οποία είναι απλώς η ανάπτυξη Taylor γύρω από το $x = 0$. Για παράδειγμα, η συνάρτηση ημίτονο έχει την εξής ανάπτυξη Maclaurin: $sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \cdots$. Αυτή η σειρά είναι αξιοσημείωτη, καθώς συγκλίνει για όλες τις τιμές του x, γεγονός που επιτρέπει την προσέγγιση του ημιτόνου με μεγάλη ακρίβεια, ακόμη και αν ληφθούν υπόψη μόνο λίγοι όροι.

Επιπλέον, οι σειρές Taylor έχουν εφαρμογές στις αριθμητικές μεθόδους, καθώς μπορούν να χρησιμοποιηθούν για την προσέγγιση συναρτήσεων όταν η άμεση αναλυτική έκφραση είναι δύσκολη ή αδύνατη. Για παράδειγμα, για να υπολογιστεί το $sin(10^\circ)$, πρώτα μετατρέπουμε τη γωνία σε ακτίνια: $10^\circ = \frac{\pi}{18} \approx \frac{1}{6}$. Συνεπώς, ακόμη και με λίγους όρους, η προσέγγιση είναι εξαιρετικά ακριβής.

Επιπρόσθετα, οι σειρές Taylor χρησιμοποιούνται στην προσέγγιση ριζικών συναρτήσεων. Για παράδειγμα, αν θέλουμε να υπολογίσουμε την τετραγωνική ρίζα του 2 με ακρίβεια πέντε δεκαδικών ψηφίων, μπορούμε να αναπτύξουμε τη

συνάρτηση $f(x) = \sqrt{x}$ γύρω από ένα κατάλληλο σημείο, όπως το $x = \frac{9}{4}$. Με αυτόν τον τρόπο, η σύγκλιση της σειράς είναι ταχύτερη και αποδίδει πολύ ακριβή τιμή για $\sqrt{2}$.

Αξίζει επίσης να αναφερθεί ότι οι σειρές Taylor επιτρέπουν την απόδειξη σημαντικών μαθηματικών θεωρημάτων. Ένα χαρακτηριστικό παράδειγμα είναι ο τύπος του Euler: $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. Αυτό το αποτέλεσμα αποδεικνύεται μέσω της ανάπτυξης της συνάρτησης $\frac{\sin(\pi x)}{\pi x}$ σε άπειρο γινόμενο και της σύγκρισης της με την αντίστοιχη σειρά Taylor. Συνεπώς, οι δυναμοσειρές δεν είναι μόνο ένα εργαλείο προσέγγισης, αλλά έχουν και θεμελιώδη ρόλο στη μαθηματική ανάλυση.

Επιπλέον, οι σειρές Taylor επιτρέπουν την εξέταση ιδιοτήτων συναρτήσεων, όπως η διαφορισιμότητα και η συνέχεια. Πράγματι, αν μια συνάρτηση μπορεί να αναπτυχθεί σε σειρά Taylor γύρω από ένα σημείο, τότε είναι άπειρα παραγωγίσιμη εκεί. Ωστόσο, υπάρχουν και εξαιρέσεις, όπως η συνάρτηση $e^{-\frac{1}{x^2}}$, η οποία έχει άπειρες παράγωγους στο $x = 0$, , αλλά η σειρά Taylor της συγκλίνει παντού στο μηδέν, χωρίς να αντιστοιχεί στη συνάρτηση.

Τέλος, οι σειρές Taylor παρέχουν έναν αποτελεσματικό τρόπο απόδειξης της αρρητότητας του αριθμού e. Αν υποθέσουμε, για χάρη της αντίφασης, ότι το e μπορεί να γραφτεί ως λόγος δύο ακεραίων, τότε χρησιμοποιώντας την ανάπτυξη Maclaurin του e^x μπορούμε να καταλήξουμε σε μια αριθμητική αντίφαση. Επομένως, μέσω αυτής της τεχνικής μπορούμε να αποδείξουμε ότι το e είναι πράγματι άρρητος αριθμός, κάτι που έχει σημαντικές επιπτώσεις στη θεωρία αριθμών.

- $\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$ for $|x| < 1$ (*Geometric Series*).
- $\ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^n}{n}$ for $|x| < 1$.
- $(1+x)^p = \sum_{n=0}^{\infty} \frac{p(p-1)\cdots(p-n+1)}{n!} x^n$ for $|x| < 1$ (*Binomial Series*).
- $\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ for all x.
- $\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$ for all x.
- $\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$ for all x.

Σχήμα 3.5: Συνηθισμένες σειρές Taylor [22]

3.4 Μέθοδος Euler

Η μέθοδος Euler [23] αποτελεί μία από τις πιο θεμελιώδεις και απλές αριθμητικές τεχνικές για την προσέγγιση λύσεων αρχικών τιμών διαφορικών εξισώσεων. Αρχικά, αναπτύχθηκε από τον Leonhard Euler το 1768 και βασίζεται στη διακριτοποίηση του συνεχούς προβλήματος, επιτρέποντας την προσέγγιση της λύσης μέσω μιας ακολουθίας σημείων που υπολογίζονται βήμα προς βήμα. Με άλλα λόγια, η μέθοδος αυτή χρησιμοποιεί ένα απλό αλγεβρικό βήμα για την προσέγγιση της τιμής της λύσης σε κάθε χρονική στιγμή, αντί να απαιτεί αναλυτική λύση της διαφορικής εξίσωσης. Ωστόσο, παρά τη σχετική απλότητά της, η μέθοδος Euler επηρεάζεται έντονα από το μέγεθος του βήματος διακριτοποίησης, γεγονός που την καθιστά επιρρεπή σε σφάλματα αποκοπής και στρογγυλοποίησης.

Η αριθμητική προσέγγιση που υιοθετεί η μέθοδος Euler βασίζεται στην ιδέα της τοπικής γραμμικοποίησης της λύσης μιας διαφορικής εξίσωσης. Πιο συγκεκριμένα, χρησιμοποιεί την εξίσωση:

$$y(t+h) \approx y(t) + hf(t, y(t))$$

όπου το νέο σημείο $y(t+h)$ υπολογίζεται μέσω της κλίσης της διαφορικής εξίσωσης στο προηγούμενο σημείο. Δηλαδή, η προσέγγιση αυτή στηρίζεται στην υπόθεση ότι η συνάρτηση παραμένει σχεδόν σταθερή σε μικρά χρονικά διαστήματα. Εντούτοις, η ακρίβεια της μεθόδου περιορίζεται από το γεγονός ότι χρησιμοποιεί μόνο τον πρώτο όρο της σειράς Taylor, αγνοώντας τους ανώτερους όρους, με αποτέλεσμα το σφάλμα της να είναι ανάλογο του βήματος διακριτοποίησης h. Συνεπώς, η ακρίβεια της μεθόδου βελτιώνεται σημαντικά όταν το h είναι πολύ μικρό, αλλά αυτό αυξάνει το συνολικό κόστος των υπολογισμών.

Αξίζει να αναφερθεί ότι η μέθοδος Euler παρουσιάζει σοβαρά προβλήματα σταθερότητας, ιδιαίτερα όταν εφαρμόζεται σε άκαμπτα συστήματα ή σε εξισώσεις με γρήγορα μεταβαλλόμενες λύσεις. Για παράδειγμα, αν η διαφορική εξίσωση περιέχει όρους που αυξάνονται εκθετικά, η μέθοδος μπορεί να δώσει ανακριβείς και ασταθείς προσεγγίσεις. Επιπλέον, η σταθερότητα εξαρτάται από το μέγεθος του βήματος h, το οποίο πρέπει να είναι αρκετά μικρό ώστε να διατηρηθεί η αριθμητική ακρίβεια, όμως όχι τόσο μικρό ώστε να αυξάνει υπερβολικά το κόστος των υπολογισμών.

Μια σημαντική επέκταση της μεθόδου Euler είναι η οπισθόδρομη (backward) μέθοδος Euler, η οποία επιλύει το πρόβλημα της σταθερότητας επιτρέποντας τη χρήση μεγαλύτερων βημάτων. Ειδικότερα, αντί να χρησιμοποιεί μόνο τη συνάρτηση $f(t+y)$ στο τρέχον σημείο, η οπισθόδρομη μέθοδος χρησιμοποιεί την τιμή της στο επόμενο σημείο, λύνοντας έτσι μια μη γραμμική εξίσωση για την πρόβλεψη της νέας τιμής. Ως αποτέλεσμα, η μέθοδος αυτή προσφέρει αυξημένη σταθερότητα όμως, απαιτεί μεγαλύτερους υπολογιστικούς πόρους λόγω της ανάγκης επίλυσης των μη γραμμικών εξισώσεων σε κάθε βήμα.

Επιπλέον, ένα άλλο σημαντικό πρόβλημα της μεθόδου Euler είναι η επίδραση των σφαλμάτων στρογγυλοποίησης και αποκοπής. Με άλλα λόγια, η χρήση πεπερασμένων βημάτων προκαλεί απόκλιση από την πραγματική λύση, και καθώς οι υπολογισμοί συσσωρεύονται, το συνολικό σφάλμα μπορεί να γίνει σημαντικό. Επιπρόσθετα, όταν η μέθοδος Euler χρησιμοποιείται σε μη γραμμικά προβλήματα ή σε προβλήματα που περιλαμβάνουν ασυμπτωτικές λύσεις, μπορεί να αποτύχει να αποδώσει ικανοποιητικά αποτελέσματα, ειδικότερα εάν το αρχικό βήμα δεν είναι σωστά ρυθμισμένο.

Για την αντιμετώπιση των περιορισμών της μεθόδου Euler, έχουν αναπτυχθεί πιο προηγμένες τεχνικές, όπως η τροποποιημένη μέθοδος Euler και οι μέθοδοι Runge-Kutta. Ειδικότερα, η τροποποιημένη μέθοδος Euler χρησιμοποιεί ένα ενδιάμεσο βήμα υπολογισμού για τη μείωση του σφάλματος και τη βελτίωση της ακρίβειας. Αντίθετα, οι μέθοδοι Runge-Kutta χρησιμοποιούν πολλαπλά ενδιάμεσα σημεία και στάδια για την προσέγγιση της λύσης, προσφέροντας υψηλότερη ακρίβεια χωρίς να αυξάνουν υπερβολικά το κόστος των υπολογισμών. Συνεπώς, οι μέθοδοι αυτές επιτρέπουν τη χρήση μεγαλύτερων βημάτων διακριτοποίησης χωρίς να θυσιάζεται η ακρίβεια, καθιστώντας τις κατάλληλες για πρακτικές εφαρμογές σε μηχανική, φυσική και υπολογιστικά μαθηματικά.

Παρ’ όλα αυτά, η μέθοδος Euler παραμένει σημαντική στη διδασκαλία αριθμητικών μεθόδων λόγω της απλότητάς της και της άμεσης εφαρμογής της σε προβλήματα αρχικών τιμών. Επιπλέον, παρέχει μια εισαγωγή στις αριθμητικές τεχνικές επίλυσης διαφορικών εξισώσεων και λειτουργεί ως θεμέλιο για πιο σύνθετες μεθόδους. Αξίζει να σημειωθεί ότι η κατανόηση των αδυναμιών της επιτρέπει στους ερευνητές να αναπτύξουν πιο αποδοτικούς και σταθερούς αλγορίθμους, βελτιώνοντας έτσι την ακρίβεια και την αποτελεσματικότητα των αριθμητικών λύσεων.

Συμπερασματικά, η μέθοδος Euler αποτελεί ένα από τα πιο βασικά εργαλεία στις αριθμητικές προσεγγίσεις των διαφορικών εξισώσεων. Αν και η απλότητα της την καθιστά ιδανική για εκπαιδευτικούς σκοπούς, οι περιορισμοί της, όπως το σφάλμα αποκοπής και η έλλειψη σταθερότητας, επιβάλλουν τη χρήση πιο εξελιγμένων μεθόδων σε πραγματικές εφαρμογές. Επομένως, η κατανόηση της μεθόδου αποτελεί θεμελιώδη βάση για την ανάπτυξη και εφαρμογή πιο ακριβών αριθμητικών τεχνικών, διαμορφώνοντας έτσι τη βάση για την αριθμητική επίλυση διαφορικών εξισώσεων.

3.5 Μέθοδοι Runge Kutta

Οι μέθοδοι Runge-Kutta [24] αναπτύχθηκαν στα τέλη του 19ου αιώνα με σκοπό τη βελτίωση της ακρίβειας των αριθμητικών μεθόδων για την προσέγγιση λύσεων διαφορικών εξισώσεων. Αρχικά, ο Carl Runge το 1895 επέκτεινε τη μέθοδο του Euler, η οποία βασιζόταν στην προώθηση της λύσης ενός αρχικού προβλήματος τιμής με διαδοχικά μικρά βήματα. Ειδικότερα, η μέθοδος Euler χρησιμοποιεί την

κλίση της λύσης μόνο στην αρχή κάθε βήματος, με αποτέλεσμα να έχει σχετικά χαμηλή ακρίβεια. Αντίθετα, ο Runge εισήγαγε τη χρήση ενδιάμεσων σημείων στο χρονικό βήμα, επιτρέποντας τη βελτίωση της ακρίβειας της προσέγγισης. Στη συνέχεια, ο Wilhelm Kutta, το 1901, γενίκευσε περαιτέρω αυτή την ιδέα, διαμορφώνοντας τις βάσεις των σύγχρονων μεθόδων Runge-Kutta.

Η μελέτη της ακρίβειας των μεθόδων Runge-Kutta εξελίχθηκε σταδιακά με τη συμβολή σημαντικών μαθηματικών, όπως οι Heun, Kutta και Nyström. Για παράδειγμα, ο Heun το 1900 ανέπτυξε μια μέθοδο τρίτης τάξης, ενώ ο Kutta εισήγαγε μεθόδους έως πέμπτης τάξης. Ωστόσο, η πραγματική επέκταση των μεθόδων αυτών πραγματοποιήθηκε το 1925 από τον Nyström, ο οποίος τις προσαρμόσε για συστήματα διαφορικών εξισώσεων δεύτερης τάξης. Αυτό ήταν ιδιαίτερα σημαντικό, καθώς επέτρεψε την επίλυση δυναμικών προβλημάτων χωρίς τη μετατροπή τους σε ισοδύναμα συστήματα πρώτης τάξης. Επομένως, οι μέθοδοι Runge-Kutta άρχισαν να βρίσκουν εφαρμογή σε ένα ευρύτερο σύνολο προβλημάτων, ιδιαίτερα στη μηχανική και τη φυσική.

Επιπλέον, η ανάγκη για αριθμητικές μεθόδους υψηλής ακρίβειας και σταθερότητας οδήγησε στη διάκριση των μεθόδων Runge-Kutta σε ρητές και άρρητες. Συγκεκριμένα, οι ρητές μέθοδοι, αν και υπολογιστικά αποδοτικές, έχουν περιορισμένη περιοχή ευστάθειας και δεν είναι κατάλληλες για άκαμπτα προβλήματα. Αντιθέτως, οι άρρητες μέθοδοι, όπως αυτές που πρότειναν οι Butcher και Kuntzmann, επιτυγχάνουν υψηλότερη σταθερότητα, αν και απαιτούν μεγαλύτερο υπολογιστικό κόστος. Ιδιαιτέρως, οι μέθοδοι Gauss-Legendre προσφέρουν A-stability, γεγονός που τις καθιστά κατάλληλες για δύσκολες περιπτώσεις διαφορικών εξισώσεων. Συνεπώς, η επιλογή της κατάλληλης μεθόδου εξαρτάται από τη φύση του προβλήματος και τις απαιτήσεις υπολογιστικής απόδοσης.

Ένα άλλο σημαντικό ζήτημα στη χρήση των μεθόδων Runge-Kutta είναι η σταθερότητα τους, καθώς η λύση των διαφορικών εξισώσεων μπορεί να είναι αριθμητικά ασταθής για συγκεκριμένες τιμές των βημάτων ολοκλήρωσης. Με άλλα λόγια, η σταθερότητα επηρεάζεται από την επιλογή της μεθόδου και το μέγεθος του χρονικού βήματος. Για τον λόγο αυτό, η έννοια της A-stability εισήχθη ως κριτήριο ευστάθειας για άρρητες μεθόδους, διασφαλίζοντας τη σταθερότητα ανεξάρτητα από την τιμή του χρονικού βήματος. Επιπρόσθετα, οι έννοιες της B-stability και algebraic stability είναι κρίσιμες για τη διατήρηση ιδιοτήτων όπως η διατήρηση της ενέργειας σε Hamiltonian συστήματα, κάτι που έχει εφαρμογές στη δυναμική των μηχανικών συστημάτων.

Τέλος, ένα κρίσιμο χαρακτηριστικό των μεθόδων Runge-Kutta είναι η εκτίμηση του σφάλματος και η δυνατότητα συνεχούς εξόδου. Για παράδειγμα, ο Merson πρότεινε πρώτος τη χρήση δύο μεθόδων διαφορετικής τάξης μέσα στο ίδιο χρονικό βήμα, ώστε να επιτρέπεται η εκτίμηση του σφάλματος. Αν και η προσέγγισή του είχε περιορισμούς, αποτέλεσε τη βάση για νεότερες τεχνικές, όπως αυτές των Fehlberg και Dormand-Prince, που επέτρεψαν τη δυναμική

προσαρμογή του βήματος ολοκλήρωσης. Επιπλέον, η δυνατότητα συνεχούς εξόδου μέσω παρεμβολής βελτίωσε σημαντικά την ακρίβεια σε εφαρμογές που απαιτούν υπολογισμό λύσεων σε ενδιάμεσα χρονικά σημεία, όπως στον έλεγχο συστημάτων και τις προσομοιώσεις φυσικών διεργασιών.

3.5.1 Μέθοδος Runge Kutta 1ης τάξης

Η μέθοδος Runge-Kutta 1ου βαθμού [25], επίσης γνωστή ως μέθοδος του Euler, είναι μια από τις πιο απλές και θεμελιώδεις αριθμητικές μεθόδους για την επίλυση ΣΔΕ. Αν και δεν είναι η πιο ακριβής μέθοδος για την επίλυση αυτών των εξισώσεων, χρησιμοποιείται ευρέως λόγω της απλότητας και της ευκολίας εφαρμογής της, ενώ παρέχει μια πρώτη προσέγγιση της λύσης σε προβλήματα όπου η υπολογιστική ισχύς ή η ακρίβεια δεν είναι η κύρια ανησυχία.

Η βασική αρχή της μεθόδου Runge-Kutta 1ου βαθμού είναι η προσέγγιση της λύσης μιας διαφορικής εξίσωσης μέσω της εξίσωσης της γραμμής της εφαπτομένης της λύσης στο κάθε σημείο του διαστήματος ολοκλήρωσης. Η μέθοδος του Euler υπολογίζει την επόμενη τιμή της λύσης σε κάθε σημείο χρησιμοποιώντας την τιμή της κλίσης της συνάρτησης στη σημερινή κατάσταση. Αυτή η μέθοδος είναι μία από τις πρώτες προσπάθειες για την αριθμητική προσέγγιση της λύσης συνηθισμένων διαφορικών εξισώσεων και είναι ιδιαίτερα κατάλληλη για την επίλυση απλών, ευθύγραμμων δυναμικών συστημάτων.

Η εξίσωση της μεθόδου του Euler για την υπολογιστική εκτίμηση της λύσης της διαφορικής εξίσωσης: $y'(x) = f\left(x, y(x)\right)$ είναι δεδομένη από:

$$y_{n+1} = y_n + h * f(x_n, y_n)$$

Όπου:

- y_n είναι η εκτίμηση της λύσης στο σημείο x_n
- h είναι το μέγεθος του βήματος ολοκλήρωσης
- $f(x_n, y_n)$ είναι η κλίση της συνάρτησης, δηλαδή η παράγωγος της y_n στο x_n

Ενώ η μέθοδος του Euler είναι πολύ εύκολη στην εφαρμογή της, έχει περιορισμένη ακρίβεια και συχνά αποτυγχάνει να δώσει αξιόπιστα αποτελέσματα σε προβλήματα με γρήγορες ή μη γραμμικές δυναμικές.

Η ακρίβεια της μεθόδου του Euler περιορίζεται από το γεγονός ότι χρησιμοποιεί μόνο την τιμή της συνάρτησης στο τρέχον σημείο για την πρόβλεψη της επόμενης τιμής, χωρίς να λαμβάνει υπόψη την κλίση στη μεσαία περιοχή του διαστήματος ολοκλήρωσης. Ως αποτέλεσμα, η μέθοδος συχνά υποτιμά τη συμπεριφορά της λύσης, ιδιαίτερα όταν η λύση της εξίσωσης παρουσιάζει γρήγορες αλλαγές ή έχει υψηλή καμπυλότητα. Οι μέθοδοι υψηλότερων βαθμών Runge-Kutta προσφέρουν μεγαλύτερη ακρίβεια, αλλά η μέθοδος του Euler

παραμένει δημοφιλής για απλές εφαρμογές λόγω της χαμηλής υπολογιστικής της πολυπλοκότητας.

Ειδικότερα, η μέθοδος του Euler είναι αρκετά χρήσιμη όταν το πρόβλημα έχει μικρή χρονική διάρκεια ή όταν οι απαιτήσεις ακρίβειας δεν είναι υψηλές. Για παράδειγμα, μπορεί να χρησιμοποιηθεί σε εφαρμογές όπως η προσομοίωση δυναμικών συστημάτων με αργή μεταβολή ή στην εκπαίδευση για την κατανόηση βασικών αριθμητικών μεθόδων. Η ευκολία της εφαρμογής της καθιστά την μέθοδο Runge-Kutta 1ου βαθμού πολύ δημοφιλή σε εκπαιδευτικά περιβάλλοντα, δίνοντας στους φοιτητές και ερευνητές μια απλή εισαγωγή στην αριθμητική λύση διαφορικών εξισώσεων.

Αξιοσημείωτο είναι ότι η μέθοδος του Euler έχει εφαρμογές σε διάφορους τομείς, από τη φυσική έως τη βιολογία και την οικονομία, αν και συνήθως χρησιμοποιείται για προβλήματα όπου η ακρίβεια της λύσης δεν είναι κρίσιμη. Σε πιο σύνθετες εφαρμογές, όπου απαιτείται μεγαλύτερη ακρίβεια, οι μέθοδοι υψηλότερου βαθμού, όπως οι Runge-Kutta 4ου βαθμού, προτιμώνται λόγω της καλύτερης απόδοσης τους όσον αφορά την ακρίβεια και τη σύγκλιση.

Ωστόσο, η μέθοδος Runge-Kutta 1ου βαθμού συνεχίζει να είναι ένα σημαντικό εργαλείο στην αριθμητική ανάλυση, και παρόλο που αντικαθίσταται από πιο προηγμένες μεθόδους σε πολλές εφαρμογές, παραμένει θεμελιώδης για την κατανόηση των βασικών αρχών της αριθμητικής επίλυσης διαφορικών εξισώσεων. Οι ερευνητές συχνά ξεκινούν με τη μέθοδο του Euler για να εκτιμήσουν τη λύση και στη συνέχεια εξετάζουν πιο ακριβείς μεθόδους για τη βελτίωση της υπολογιστικής απόδοσης και της ακρίβειας της λύσης τους.

3.5.2 Μέθοδος Runge Kutta 2^{ης} τάξης

Η μέθοδος Runge-Kutta 2ου βαθμού [25], γνωστή επίσης ως μέθοδος Heun, αποτελεί μια βελτιωμένη έκδοση της μεθόδου του Euler για την αριθμητική επίλυση ΣΔΕ. Αν και η μέθοδος του Euler χρησιμοποιεί μόνο την τιμή της συνάρτησης στο τρέχον σημείο για την εκτίμηση της λύσης στο επόμενο βήμα, η μέθοδος του Heun (Runge-Kutta 2ου βαθμού) χρησιμοποιεί μια προσέγγιση που λαμβάνει υπόψη τόσο την τιμή της συνάρτησης στο αρχικό όσο και στο τελικό σημείο του διαστήματος ολοκλήρωσης. Αυτή η βελτίωση οδηγεί σε καλύτερη ακρίβεια σε σχέση με τη μέθοδο του Euler, χωρίς να απαιτεί σημαντική αύξηση στην υπολογιστική πολυπλοκότητα.

Η βασική ιδέα πίσω από τη μέθοδο Runge-Kutta 2ου βαθμού είναι να υπολογίσει μια ενδιάμεση εκτίμηση για την επόμενη τιμή της λύσης, χρησιμοποιώντας ένα μέσο όρο των τιμών της συνάρτησης στο αρχικό και στο τελικό σημείο του βήματος ολοκλήρωσης. Αυτός ο μέσος όρος παρέχει μια πιο ακριβή εκτίμηση της κλίσης, αποφεύγοντας την υπερεκτίμηση ή την υποεκτίμηση της μεταβολής της λύσης που συμβαίνει στη μέθοδο του Euler. Η εξίσωση της

μεθόδου Runge-Kutta 2ου βαθμού για την εκτίμηση της λύσης της διαφορικής εξίσωσης: $y'(x) = f\left(x, y(x)\right)$ είναι η εξής:

$$\mathrm{k1} = h * f(x_n, y_n)$$

$$\mathrm{k2} = h * f(x_n + h, y_n + k1)$$

$$y_{n+1} = y_n + \frac{1}{2}(k1 + k2)$$

Όπου:

- $k1$ είναι η εκτίμηση της κλίσης στο αρχικό σημείο του βήματος
- $k2$ είναι η εκτίμηση της κλίσης στο τελικό σημείο του βήματος
- h είναι το μέγεθος του βήματος ολοκλήρωσης
- y_n είναι η εκτίμηση της λύσης στο σημείο x_n
- $y_n + 1$ είναι η εκτίμηση της λύσης στο επόμενο σημείο $x_n + h$

Η μέθοδος αυτή αποτελεί έναν ενδιάμεσο συμβιβασμό μεταξύ της απλότητας της μεθόδου του Euler και της υψηλότερης ακρίβειας που επιτυγχάνεται με τις μεθόδους Runge-Kutta 4ου βαθμού. Χρησιμοποιώντας δύο εκτιμήσεις της κλίσης σε κάθε βήμα, η μέθοδος Runge-Kutta 2ου βαθμού προσφέρει σημαντική βελτίωση της ακρίβειας, ενώ διατηρεί την υπολογιστική της απόδοση.

Η μέθοδος Heun είναι ιδιαίτερα χρήσιμη σε προβλήματα όπου απαιτείται υψηλότερη ακρίβεια από τη μέθοδο του Euler, αλλά δεν είναι απαραίτητο να χρησιμοποιηθούν οι πιο πολύπλοκες μεθόδους 4ου βαθμού. Η επιπλέον υπολογιστική πολυπλοκότητα σε σύγκριση με τη μέθοδο του Euler είναι περιορισμένη, καθώς απαιτεί μόνο τον υπολογισμό ενός δεύτερου "κ" όρου ($k2$). Ωστόσο, η αύξηση της ακρίβειας είναι σημαντική, καθώς μειώνεται ο αριθμός των σφαλμάτων που εισάγονται από την υποεκτίμηση ή υπερεκτίμηση της κλίσης της συνάρτησης.

Αναλυτικά, η μέθοδος Runge-Kutta 2ου βαθμού είναι μια καλή επιλογή για προβλήματα όπου απαιτείται μεσαία ακρίβεια και δεν είναι απαραίτητη η πολύπλοκη εφαρμογή πιο εξελιγμένων μεθόδων. Μπορεί να χρησιμοποιηθεί για την επίλυση διαφορικών εξισώσεων με σχετικά σταθερές ή μέτρια γρήγορες αλλαγές, ενώ παράλληλα προσφέρει καλύτερη απόδοση σε σχέση με τη μέθοδο του Euler. Παράλληλα, η μέθοδος του Heun συχνά χρησιμοποιείται ως πρότυπο στην εκπαίδευση, καθώς παρέχει μια φυσική συνέχεια στην ανάπτυξη αριθμητικών μεθόδων και διευκολύνει τη μετάβαση σε πιο προηγμένες τεχνικές, όπως οι μέθοδοι Runge-Kutta 4ου βαθμού.

Η ακριβέστερη εκτίμηση των τιμών της λύσης καθιστά τη μέθοδο αυτή κατάλληλη για ευρύ φάσμα εφαρμογών, από τη μηχανική και τη φυσική έως τη βιολογία και τις κοινωνικές επιστήμες, και χρησιμοποιείται συχνά για την

προσομοίωση συστημάτων όπου η απόδοση και η ακρίβεια είναι σημαντικά, αλλά η υπολογιστική πολυπλοκότητα πρέπει να παραμένει σε χαμηλά επίπεδα. Για παράδειγμα, η μέθοδος μπορεί να χρησιμοποιηθεί στην προσομοίωση ρευστών, στη μελέτη δυναμικών συστημάτων ή στην ανάλυση βιολογικών συστημάτων.

Συνοψίζοντας, η μέθοδος Runge-Kutta 2ου βαθμού προσφέρει έναν αποτελεσματικό και ευέλικτο τρόπο για την επίλυση διαφορικών εξισώσεων, παρέχοντας καλύτερη ακρίβεια από τη μέθοδο του Euler χωρίς να απαιτεί υπερβολική υπολογιστική ισχύ. Αν και μπορεί να μην έχει την ακρίβεια των πιο σύνθετων μεθόδων, η ευκολία εφαρμογής της και η αναλογία υπολογιστικού κόστους/ακρίβειας την καθιστούν ιδιαίτερα χρήσιμη σε πολλές εφαρμογές.

3.5.3 Μέθοδος Runge Kutta 3ης τάξης

Η μέθοδος Runge-Kutta 3ου βαθμού [25] είναι μια πιο εξελιγμένη μέθοδος σε σχέση με τις μεθόδους 1ου και 2ου βαθμού, προσφέροντας μεγαλύτερη ακρίβεια για την επίλυση ΣΔΕ χωρίς να αυξάνει υπερβολικά την υπολογιστική πολυπλοκότητα. Αυτή η μέθοδος ενσωματώνει τρία στάδια υπολογισμού της κλίσης της συνάρτησης, προσφέροντας μια βελτιωμένη εκτίμηση της λύσης σε κάθε βήμα, κάτι που την καθιστά πιο ακριβή και ισχυρή από τη μέθοδο του Euler και τη μέθοδο Runge-Kutta 2ου βαθμού.

Η βασική ιδέα της μεθόδου Runge-Kutta 3ου βαθμού είναι να υπολογίσει την τιμή της λύσης της διαφορικής εξίσωσης μέσω τριών διαφορετικών εκτιμήσεων της κλίσης σε κάθε βήμα. Αυτές οι εκτιμήσεις υπολογίζονται μέσω τριών διαφορετικών τιμών $k1$, $k2$ & $k3$, οι οποίες στη συνέχεια συνδυάζονται με βάση κάποιο συγκεκριμένο βάρος για την παραγωγή της τελικής εκτίμησης της επόμενης τιμής της λύσης. Η μέθοδος αυτή προτείνει τα εξής βήματα υπολογισμού:

$$\mathrm{k1} = h * f(x_n, y_n)$$

$$\mathrm{k2} = h * f\left(x_n + \frac{h}{2}, y_n + \frac{k1}{2}\right)$$

$$\mathrm{k3} = h * f(x_n + h,\ y_n - \ k1\ +\ 2k2)$$

$$y_{n+1} = y_n + \frac{1}{6}(k1\ +\ 4k2\ +\ k3)$$

Όπου:

- $k1$, $k2$ & $k3$ είναι οι εκτιμήσεις της κλίσης της συνάρτησης στη διαδικασία των τριών υπολογιστικών σταδίων
- $f(x_n, y_n)$ είναι η κλίση της συνάρτησης, δηλαδή η παράγωγος της y_n στο x_n
- h είναι το μέγεθος του βήματος ολοκλήρωσης
- y_n είναι η εκτίμηση της λύσης στο σημείο x_n
- $y_n + 1$ είναι η εκτίμηση της λύσης στο επόμενο σημείο $x_n + h$

Η μέθοδος Runge-Kutta 3ου βαθμού προσφέρει βελτιωμένη ακρίβεια σε σύγκριση με τη μέθοδο 2ου βαθμού, χρησιμοποιώντας περισσότερες εκτιμήσεις της κλίσης, ενώ ταυτόχρονα διατηρεί μια λογική υπολογιστική πολυπλοκότητα. Το βάρος που αποδίδεται στις διάφορες εκτιμήσεις της κλίσης (δηλαδή, το βάρος $\frac{1}{6}$ για $k1$, $\frac{4}{6}$ για $k2$ & $\frac{1}{6}$ για $k3$) επιτρέπει στη μέθοδο να αξιοποιήσει καλύτερα τις διακυμάνσεις της κλίσης και να μειώσει τα σφάλματα που μπορεί να εισαχθούν μέσω των πρώτων εκτιμήσεων.

Η μεγαλύτερη ακρίβεια που επιτυγχάνεται με τη μέθοδο αυτή την καθιστά κατάλληλη για πιο απαιτητικά προβλήματα, όπου η λύση της διαφορικής εξίσωσης δεν είναι γραμμική ή όταν η συνάρτηση $f(x_n, y_n)$ παρουσιάζει έντονες αλλαγές κατά τη διάρκεια της ολοκλήρωσης. Η μέθοδος Runge-Kutta 3ου βαθμού είναι επίσης πιο ανθεκτική σε προβλήματα όπου το μέγεθος του βήματος h δεν είναι αυστηρά μικρό και παρέχει μια καλύτερη εκτίμηση σε σχέση με τις μεθόδους 1ου και 2ου βαθμού, μειώνοντας τα σφάλματα σε μεγαλύτερα βήματα.

Ένα από τα πλεονεκτήματα της μεθόδου αυτής είναι η ευελιξία της. Καθώς η μέθοδος απαιτεί μόνο τρία βήματα για την εκτίμηση της λύσης σε κάθε βήμα ολοκλήρωσης, παραμένει σχετικά απλή στην εφαρμογή της, ενώ ταυτόχρονα παρέχει σημαντική βελτίωση στην ακρίβεια των υπολογισμών. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη σε περιοχές εφαρμογής που απαιτούν υψηλότερη ακρίβεια από τις μεθόδους πρώτου και δεύτερου βαθμού, όπως η μηχανική, η φυσική, η βιολογία και τα οικονομικά μοντέλα.

Η μέθοδος Runge-Kutta 3ου βαθμού εφαρμόζεται ευρέως σε τομείς όπως η δυναμική συστημάτων, η ανάλυση ρευστών, οι βιολογικές διεργασίες και η προσομοίωση των φυσικών φαινομένων. Για παράδειγμα, μπορεί να χρησιμοποιηθεί στην προσομοίωση της κίνησης σωμάτων υπό βαρυτική έλξη, στην ανάλυση της μετάδοσης κυμάτων ή στη μελέτη του μεταβολισμού των οργανισμών.

Παρ' όλο που η μέθοδος αυτή έχει καλύτερη ακρίβεια από τις μεθόδους 1ου και 2ου βαθμού, δεν φτάνει στην ακρίβεια των μεθόδων υψηλότερων βαθμών, όπως η μέθοδος Runge-Kutta 4ου βαθμού. Ωστόσο, η εφαρμογή της είναι συχνά επιθυμητή σε περιπτώσεις όπου η βελτίωση της ακρίβειας είναι σημαντική αλλά δεν δικαιολογείται η αυξημένη υπολογιστική πολυπλοκότητα της μεθόδου 4ου βαθμού.

Συνοψίζοντας, η μέθοδος Runge-Kutta 3ου βαθμού προσφέρει μια εξαιρετική ισορροπία μεταξύ υπολογιστικής απόδοσης και ακρίβειας. Χρησιμοποιώντας τρεις εκτιμήσεις της κλίσης σε κάθε βήμα, επιτυγχάνει πιο ακριβείς λύσεις από τις μεθόδους 1ου και 2ου βαθμού, κάνοντάς την κατάλληλη για πιο απαιτητικά προβλήματα. Με τη σωστή επιλογή του μεγέθους του βήματος h, η μέθοδος αυτή μπορεί να εφαρμοστεί σε ένα ευρύ φάσμα επιστημονικών και μηχανικών πεδίων,

εξασφαλίζοντας ακριβείς και αξιόπιστες εκτιμήσεις της λύσης διαφορικών εξισώσεων.

3.5.4 Μέθοδος Runge Kutta 4ης τάξης

Η μέθοδος Runge-Kutta 4ου βαθμού [25] (γνωστή και ως RK4) είναι μία από τις πιο δημοφιλείς και ευρέως χρησιμοποιούμενες μεθόδους για την επίλυση ΣΔΕ. Εξελίχθηκε για να προσφέρει έναν εξαιρετικό συνδυασμό ακρίβειας και υπολογιστικής απόδοσης, καθιστώντας την μια εξαιρετική επιλογή για πλήθος εφαρμογών στις φυσικές, βιολογικές και μηχανικές επιστήμες. Αν και οι μέθοδοι Runge-Kutta χαμηλότερων βαθμών προσφέρουν λογική υπολογιστική πολυπλοκότητα και ευκολία στην εφαρμογή τους, η RK4 ξεχωρίζει για την υψηλή της ακρίβεια χωρίς την ανάγκη αυξημένων υπολογιστικών πόρων σε σχέση με μεθόδους υψηλότερων βαθμών.

Η βασική ιδέα πίσω από τη μέθοδο Runge-Kutta 4ου βαθμού είναι η εκτίμηση της λύσης μιας διαφορικής εξίσωσης μέσω τεσσάρων υπολογισμών της κλίσης (κλίση της συνάρτησης $f(x_n, y_n)$ για τα διάφορα σημεία στην περιοχή του βήματος ολοκλήρωσης). Η μέθοδος χρησιμοποιεί έναν προσεγγιστικό αλγόριθμο, που περιλαμβάνει τέσσερα στάδια υπολογισμού (εκτιμήσεις της κλίσης) για κάθε βήμα h. Ουσιαστικά, η μέθοδος συνδυάζει αυτές τις εκτιμήσεις με διάφορα βάρη για να παραχθεί η καλύτερη δυνατή εκτίμηση για την επόμενη τιμή της λύσης της εξίσωσης.

Η διαδικασία υπολογισμού στη μέθοδο Runge-Kutta 4ου βαθμού περιλαμβάνει τα εξής βήματα:

$$\mathrm{k1} = h * f(x_n, y_n)$$

$$\mathrm{k2} = h * f\left(x_n + \frac{h}{2}, y_n + \frac{k1}{2}\right)$$

$$\mathrm{k3} = h * f\left(x_n + \frac{h}{2},\ y_n + \frac{k2}{2}\right)$$

$$\mathrm{k4} = h * f(x_n + \ h,\ y_n + \ k3)$$

$$y_{n+1} = y_n + \frac{1}{6}(k1 \ + \ 2k2 \ + \ 2k3 \ + \ k4)$$

Όπου:

- $k1, k2, k3$ & $k4$ είναι οι εκτιμήσεις της κλίσης της συνάρτησης στα τέσσερα στάδια υπολογισμού, που χρησιμοποιούνται για την προσαρμογή της λύσης. Κάθε εκτίμηση βασίζεται σε διαφορετικό σημείο εντός του διαστήματος h, παρέχοντας έτσι μια πιο ακριβή προσέγγιση της μέσης κλίσης.
- $f(x_n, y_n)$ είναι η κλίση της συνάρτησης, δηλαδή η παράγωγος της y_n στο x_n

- h είναι το μέγεθος του βήματος ολοκλήρωσης
- y_n είναι η εκτίμηση της λύσης στο σημείο x_n
- $y_n + 1$ είναι η εκτίμηση της λύσης στο επόμενο σημείο $x_n + h$

Η μέθοδος αυτή προσφέρει υψηλή ακρίβεια γιατί συνδυάζει τέσσερις διαφορετικές εκτιμήσεις της κλίσης σε διάφορα σημεία του βήματος και τις συνδυάζει με βάρη για να δώσει την καλύτερη δυνατή προσεγγιστική τιμή για την επόμενη τιμή της λύσης. Αυτή η διαδικασία επιτρέπει στην μέθοδο να διατηρεί την ακρίβεια ακόμα και για μεγαλύτερα βήματα h. Σε αντίθεση με άλλες μεθόδους, όπως η μέθοδος του Euler ή του Runge-Kutta 2ου βαθμού, η RK4 παρέχει μια πιο ακριβή εκτίμηση χωρίς την ανάγκη αυξημένων υπολογιστικών πόρων, κάνοντάς την μια δημοφιλή επιλογή για τη λύση πιο σύνθετων και μη γραμμικών διαφορικών εξισώσεων.

Η μέθοδος Runge-Kutta 4ου βαθμού είναι ιδιαίτερα χρήσιμη σε προβλήματα όπου απαιτείται υψηλή ακρίβεια χωρίς να αυξάνεται υπερβολικά ο υπολογιστικός φόρτος. Η ικανότητά της να υπολογίζει με ακρίβεια την επόμενη τιμή της λύσης σε κάθε βήμα επιτρέπει τη λύση πιο σύνθετων και μη γραμμικών διαφορικών εξισώσεων, ενώ διατηρεί χαμηλό το κόστος υπολογισμού σε σχέση με πιο ακριβείς μεθόδους (π.χ., μέθοδοι υψηλότερων βαθμών). Επιπλέον, η μέθοδος RK4 είναι πολύ ευέλικτη, διότι δεν απαιτεί την εκ των προτέρων γνώση της λύσης της εξίσωσης ή κάποιου ειδικού χαρακτηριστικού της, γεγονός που την καθιστά ιδανική για την εφαρμογή σε ένα ευρύ φάσμα προβλημάτων.

Στη συνέχεια, η έρευνα μας θα επικεντρωθεί πρακτικά στην εφαρμογή της μεθόδου Runge-Kutta 4ου βαθμού, καθώς αυτή η μέθοδος αντιπροσωπεύει μια από τις πιο αξιόπιστες και αποτελεσματικές προσεγγίσεις για την επίλυση διαφορικών εξισώσεων με ακρίβεια, χωρίς να απαιτεί υπερβολική υπολογιστική ισχύ ή χρόνο.

3.6 Δυναμική Μεθόδων Runge Kutta

Οι μέθοδοι Runge Kutta [26] βασίζονται στην εκτίμηση του ολοκληρώματος μέσω μιας σειράς ενδιάμεσων υπολογισμών της παραγώγου. Ειδικότερα, η βασική τους μορφή δίνεται από την εξίσωση:

$$Y_{n+1} = Y_n + h \sum_{i=1}^{q} \omega_i \, k_i$$

όπου τα k_i υπολογίζονται μέσω της σχέσης:

$$k_i = f\left(x_n + h\alpha_i, Y_n + h \sum_{j=1}^{i-1} \beta_{ij} \, k_j \right)$$

για ρητές (implicit) μεθόδους.

Ακόμα, η ακρίβεια των μεθόδων Runge-Kutta καθορίζεται από το τοπικό σφάλμα αποκοπής, το οποίο είναι τάξης $O(h^p)$, όπου p η τάξη της μεθόδου. Η σύγκλιση της μεθόδου εξασφαλίζεται εφόσον ισχύει η συνθήκη συνοχής (consistency condition):

$$\sum_{i=1}^{q} \omega_i = 1$$

Μάλιστα, είναι ιδιαίτερα σημαντικό να αναφερθεί ότι οι ρητές μέθοδοι Runge-Kutta έχουν ανώτατο όριο τάξης το 4, δηλαδή για $p > 4$ απαιτείται η χρήση άρρητων μεθόδων.

Η αριθμητική σταθερότητα των μεθόδων αυτών εξετάζεται μέσω της ανάλυσης της απόλυτης σταθερότητας (absolute stability). Συγκεκριμένα, η περιοχή απόλυτης σταθερότητας προσδιορίζεται από την ιδιότητα: $|S| \leq 1$, όπου S είναι η συνάρτηση σταθερότητας, δηλαδή το χαρακτηριστικό πολυώνυμο της μεθόδου που εφαρμόζεται στο γραμμικό πρόβλημα $y' = \lambda y$. Στην πράξη, οι ρητές μέθοδοι έχουν πεπερασμένες περιοχές απόλυτης σταθερότητας, ενώ οι άρρητες μπορούν να έχουν εκτεταμένες ή και απεριόριστες περιοχές σταθερότητας.

Όταν οι μέθοδοι Runge-Kutta εφαρμόζονται σε άκαμπτα προβλήματα (stiff problems), εμφανίζονται σοβαρές προκλήσεις. Συγκεκριμένα, ένα πρόβλημα χαρακτηρίζεται ως άκαμπτο εάν οι ιδιοτιμές λi του Ιακωβιανού πίνακα του συστήματος έχουν μεγάλη αρνητική πραγματική τιμή, δηλαδή:

$$R = \frac{max_{1 \leq i \leq n} |\, Re(\lambda i)|}{min_{1 \leq i \leq n} |\, Re(\lambda i)|}.$$

Η δυσκολία έγκειται στο γεγονός ότι οι ρητές μέθοδοι έχουν περιορισμένες περιοχές απόλυτης σταθερότητας, με αποτέλεσμα να απαιτούν πολύ μικρά χρονικά βήματα για τη διατήρηση της σταθερότητας.

Για την αντιμετώπιση αυτού του προβλήματος, χρησιμοποιούνται άρρητες Runge-Kutta μέθοδοι, καθώς ορισμένες από αυτές είναι A-stable, δηλαδή διατηρούν τη σταθερότητα της αριθμητικής λύσης για κάθε τιμή του βήματος h. Ωστόσο, η χρήση τους απαιτεί την επίλυση μη γραμμικών συστημάτων σε κάθε βήμα, γεγονός που αυξάνει το υπολογιστικό κόστος.

Εκτός από τη γραμμική σταθερότητα, σημαντική είναι και η μη γραμμική απόλυτη σταθερότητα (nonlinear absolute stability). Η μη γραμμική σταθερότητα αναφέρεται στη συμπεριφορά των Runge-Kutta μεθόδων όταν εφαρμόζονται σε μη γραμμικά δυναμικά συστήματα, όπως η λογιστική εξίσωση: $y' = \lambda y(1 - y)$. Αξίζει να αναφερθεί ότι σε τέτοιες περιπτώσεις η περιοχή σταθερότητας μπορεί να μεταβάλλεται, ανάλογα με τις ιδιότητες του μη γραμμικού συστήματος.

Ένα άλλο ενδιαφέρον φαινόμενο που παρατηρείται στις Runge-Kutta μεθόδους είναι η εμφάνιση φανταστικών σημείων ισορροπίας (ghost fixed points), τα οποία δεν υπάρχουν στο αρχικό συνεχές σύστημα. Αυτό είναι αποτέλεσμα της διακριτοποίησης του προβλήματος και μπορεί να οδηγήσει σε ανεπιθύμητες αποκλίσεις της αριθμητικής λύσης. Συνεπώς, είναι κρίσιμο να λαμβάνεται υπόψη το μέγεθος του χρονικού βήματος, ιδιαίτερα σε χαοτικά συστήματα.

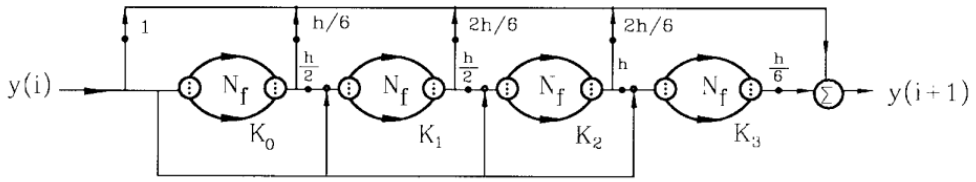
Η βελτιστοποίηση των μεθόδων Runge-Kutta για εφαρμογές μηχανικής μάθησης είναι ένα ενεργό πεδίο έρευνας. Για παράδειγμα, οι τεχνικές προσαρμογής του χρονικού βήματος (adaptive step-size control) επιτρέπουν τη δυναμική αλλαγή του h με βάση τις εκτιμήσεις σφάλματος. Χαρακτηριστικό παράδειγμα τέτοιων τεχνικών αποτελούν οι μέθοδοι Dormand-Prince και Runge-Kutta-Fehlberg (RKF).

Συμπερασματικά, οι μέθοδοι Runge-Kutta παρέχουν ισχυρά εργαλεία για την αριθμητική επίλυση διαφορικών εξισώσεων, ωστόσο η αποτελεσματικότητά τους εξαρτάται από τη φύση του προβλήματος. Η επιλογή της κατάλληλης μεθόδου απαιτεί προσεκτική ανάλυση της σταθερότητας, της ακρίβειας και του υπολογιστικού κόστους, ιδίως σε περιπτώσεις άκαμπτων και μη γραμμικών συστημάτων.

4 Σχετική Βιβλιογραφία

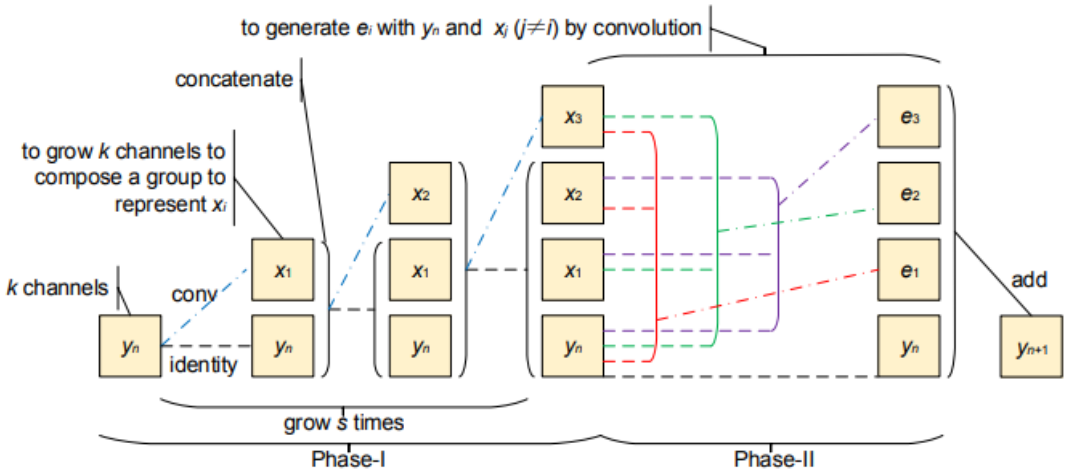
Η μέθοδος Runge-Kutta (RK4) αποτελεί ένα ισχυρό εργαλείο βελτιστοποίησης νευρωνικών δικτύων, προσφέροντας σημαντικά πλεονεκτήματα, όπως υψηλή ακρίβεια, καλύτερη γενίκευση και ενισχυμένη σταθερότητα σε δυναμικά και μη γραμμικά προβλήματα. Η διεθνής βιβλιογραφία αποδεικνύει την αποτελεσματικότητα της RK4 μέσω της ενσωμάτωσής της σε διάφορες αρχιτεκτονικές νευρωνικών δικτύων, από κλασικά feedforward μοντέλα μέχρι convolutional και residual δίκτυα.

Αρχικά, η εργασία των Wang και Lin του 1998 [1], εισάγει ως πρωτοπόρα τα Runge-Kutta Neural Networks (RKNNs), τα οποία βασίζονται στη μέθοδο RK4 για την αναγνώριση δυναμικών συστημάτων που περιγράφονται από διαφορικές εξισώσεις. Συγκεκριμένα, τα RKNNs χρησιμοποιούν ένα νευρωνικό υποδίκτυο για να προσεγγίσουν τη δεξιά πλευρά της διαφορικής εξίσωσης, βελτιώνοντας την ακρίβεια στην εκτίμηση των ρυθμών μεταβολής των συστημάτων. Αυτή η προσέγγιση επιτρέπει την εκτέλεση μακροπρόθεσμων προβλέψεων με υψηλή ακρίβεια και μικρότερα σφάλματα σε σχέση με τα παραδοσιακά δίκτυα που βασίζονται σε χαμηλής τάξης διακριτοποίηση.



Σχήμα 4.1: Αρχιτεκτονική του Νευρωνικού Δικτύου Runge-Kutta (RKNN) [1]

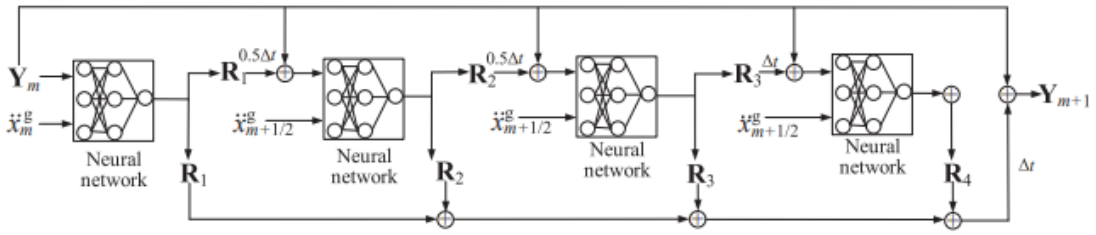
Ακόμα, οι μέθοδοι Runge-Kutta επεκτάθηκαν και στα Convolutional Neural Networks (CNNs) μέσω των RK Convolutional Neural Networks (RKCNNs) [27]. Οι θεμελιωτές αυτής της ιδέας χρησιμοποίησαν τις μεθόδους RK για να διαχειριστούν τα στάδια της εκπαίδευσης, ελαχιστοποιώντας τα σφάλματα διακριτοποίησης και επιτυγχάνοντας βελτίωση της ακρίβειας σε benchmarks ταξινόμησης εικόνων, όπως MNIST και CIFAR-10. Σε σύγκριση με τα παραδοσιακά ResNets, τα RKCNNs προσφέρουν υψηλότερη ακρίβεια με μειωμένες απαιτήσεις σε υπολογιστικούς πόρους, καθιστώντας τα ιδιαίτερα χρήσιμα για εφαρμογές όρασης υπολογιστών.



Σχήμα 4.2: Το σχήμα παρουσιάζει την αρχιτεκτονική ενός χρονικού βήματος σε ένα RKCNN-R με τη μέθοδο Runge-Kutta, όπου κάθε στάδιο δημιουργεί ένα ενδιάμεσο αποτέλεσμα που συνδυάζεται με το αρχικό για τον υπολογισμό της επόμενης τιμής. [27]

Οι βασικοί θεμελιωτές που εισήγαγαν τα Strong Stability Preserving Networks (SSPNets) [28], τα οποία βασίζονται σε Runge-Kutta μεθόδους για τη διατήρηση της σταθερότητας σε δίκτυα που δέχονται επιθέσεις από adversarial παραδείγματα. Αυτές οι μέθοδοι προσφέρουν μεγαλύτερη ακρίβεια και ανθεκτικότητα σε σχέση με τα ResNets, καθώς καταφέρνουν να καταστείλουν την ανάπτυξη θορύβου κατά την προώθηση (forward propagation). Η χρήση παραμέτρων προερχόμενων από την RK διακριτοποίηση και επιτρέπει στα SSPNets να βελτιώσουν τη σταθερότητα και την ανθεκτικότητα χωρίς αύξηση των παραμέτρων του δικτύου.

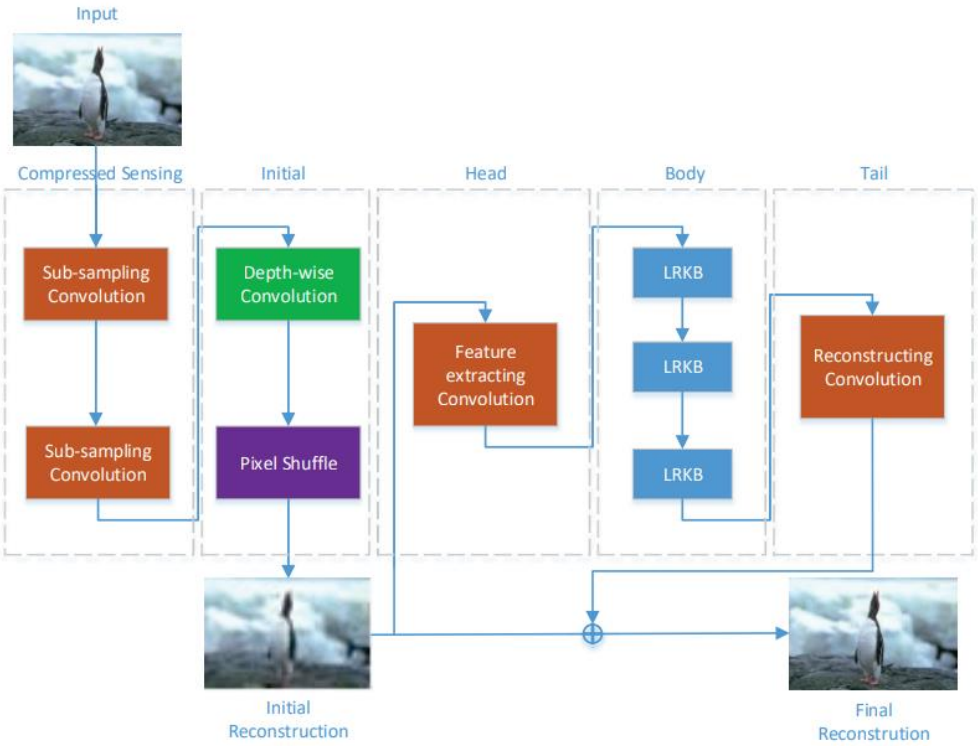
Επίσης, η εργασία [29] που προτείνει τα Runge-Kutta Recurrent Neural Networks (RKRNNs) για την πρόβλεψη σεισμικής απόκρισης σε δυναμικά μηχανικά συστήματα, αποτελεί ένα εξαιρετικό παράδειγμα της χρήσης των RKRNN σε προβλήματα του πραγματικού κόσμου. Ουσιαστικά, τα RKRNNs συνδυάζουν την παραδοσιακή προσέγγιση των RNNs με την αριθμητική μέθοδο Runge-Kutta 4ου βαθμού, που διασφαλίζει σταθερότητα και υψηλή ακρίβεια κατά την εκπαίδευση και την πρόβλεψη. Η ενσωμάτωση της RK4 σε RNNs στοχεύει στη μείωση των συσσωρευόμενων σφαλμάτων που προκύπτουν από τη μακροπρόθεσμη πρόβλεψη χρονικών σειρών. Το μοντέλο εφαρμόστηκε σε δεδομένα παρακολούθησης σεισμικής δραστηριότητας και απέδωσε καλύτερη προσαρμογή και ακρίβεια σε σχέση με τις παραδοσιακές μεθόδους. Τελικά το RKRNN αποδείχθηκε ιδανικό για προβλήματα μη γραμμικής δυναμικής, παρέχοντας χρήσιμες εφαρμογές στη μηχανική και την ανάλυση πολύπλοκων συστημάτων.



Σχήμα 4.3: Αρχιτεκτονική του RKRNN [29]

Σύμφωνα με μια νέα μελέτη [30], τα Runge-Kutta Neural Networks (RKNNs) προσφέρουν μια αποδοτική προσέγγιση για τη μείωση πολυπλοκότητας σε μεγάλα μαθηματικά μοντέλα, γνωστή ως Model Order Reduction (MOR). Αυτή η μέθοδος χρησιμοποιεί τη μέθοδο Runge-Kutta για την επίλυση διαφορικών εξισώσεων σε περιορισμένα διαστήματα χώρου, προσφέροντας ταχύτερες και πιο ακριβείς λύσεις. Συγκεκριμένα, η μελέτη δείχνει ότι τα RKNNs μπορούν να επιλύσουν πολύπλοκες ΣΔΕ που σχετίζονται με φυσικές διεργασίες, όπως τη δυναμική ρευστών ή τη θερμοδυναμική. Αυτό επιτυγχάνεται μειώνοντας την υπολογιστική πολυπλοκότητα, διατηρώντας παράλληλα τη φυσική συνέπεια του προβλήματος. Μάλιστα, ένα σημαντικό πλεονέκτημα της μεθόδου είναι η προσαρμοστικότητα και η ικανότητά της να εκπαιδεύει μοντέλα υψηλής ακρίβειας με μειωμένο αριθμό παραμέτρων.

Τέλος, μια πρόσφατη εργασία [31] παρουσιάζει τα Runge-Kutta Convolutional Compressed Sensing Networks (RK-CCSNet), τα οποία στοχεύουν στην ανακατασκευή εικόνων μέσω δυναμικών συστημάτων. Η μέθοδος αυτή ενσωματώνει μπλοκ Runge-Kutta για τη βελτίωση της ακρίβειας στην αποκατάσταση εικόνων από συμπιεσμένα δεδομένα. Το RK-CCSNet βασίζεται σε δομές CNNs, όπου τα μπλοκ Runge-Kutta διαχειρίζονται τη δυναμική των δεδομένων και βελτιώνουν την ποιότητα της ανακατασκευής. Επίσης, τα αποτελέσματα έδειξαν ότι το μοντέλο αποδίδει καλύτερα σε χαμηλούς ρυθμούς δειγματοληψίας, με λιγότερα σφάλματα σε σχέση με παραδοσιακές μεθόδους. Αυτή η προσέγγιση έχει εφαρμογές σε ιατρικές εικόνες και τεχνικές ανάλυσης σήματος.



Σχήμα 4.4: Βασική Δομή του RK-CCSNet [31]

5 Πειραματισμός & Αποτελέσματα

Σε αυτό το κεφάλαιο, παρουσιάζεται η πειραματική ανάλυση που επικεντρώνεται στην εφαρμογή της μεθόδου Runge-Kutta τέταρτης τάξης (RK4) για τη βελτιστοποίηση των παραμέτρων ενός νευρωνικού δικτύου. Πιο συγκεκριμένα, στόχος είναι να διερευνηθεί πώς η RK4 μπορεί να χρησιμοποιηθεί όχι μόνο ως αριθμητική τεχνική για την επίλυση ΠΑΤ με ΣΔΕ, αλλά και ως ένα αποτελεσματικό εργαλείο βελτιστοποίησης. Με άλλα λόγια, η μέθοδος εφαρμόζεται με σκοπό να βελτιώσει τα αποτελέσματα του νευρωνικού δικτύου, προτείνοντας μια νέα προσέγγιση στην εκπαίδευση νευρωνικών δικτύων.

Η υλοποίηση πραγματοποιείται σε περιβάλλον Python, χρησιμοποιώντας το Google Colab, καθώς προσφέρει ευελιξία και ισχυρά υπολογιστικά εργαλεία. Ειδικότερα, η πειραματική διαδικασία θα επικεντρωθεί στην αξιολόγηση της αποτελεσματικότητας της RK4 σε σχέση με καθιερωμένους αλγορίθμους βελτιστοποίησης, όπως ο Full- Batch Gradient Descent και ο Adam. Εντούτοις, σε αντίθεση με αυτές τις παραδοσιακές μεθόδους, η RK4 προσεγγίζει το πρόβλημα μέσω αριθμητικής ολοκλήρωσης, γεγονός που ενδέχεται να προσφέρει διαφορετικά πλεονεκτήματα στην ταχύτητα σύγκλισης και την ακρίβεια.

Επιπλέον, αξίζει να αναφερθεί ότι τα αποτελέσματα των πειραμάτων θα συγκριθούν με εκείνα των παραδοσιακών τεχνικών, με σκοπό να διαπιστωθεί αν η προτεινόμενη μέθοδος μπορεί να επιτύχει αντίστοιχους χρόνους εκπαίδευσης και συγκρίσιμη απόδοση. Εφόσον επιτευχθούν τέτοια αποτελέσματα, η RK4 θα μπορούσε να αποτελέσει μια πολλά υποσχόμενη εναλλακτική για τη βελτιστοποίηση νευρωνικών δικτύων. Συνεπώς, η ανάλυση που ακολουθεί έχει ιδιαίτερη σημασία, καθώς μπορεί να προσφέρει νέα προοπτική στον τρόπο με τον οποίο προσεγγίζεται η εκπαίδευση τεχνητών νευρωνικών δικτύων.

5.1 Σύγκριση μεταξύ των μεθόδων Runge Kutta (RK1/RK2/RK3/RK4)

Σε αυτήν την ενότητα, παρουσιάζεται η σύγκριση μεταξύ των διαφόρων τάξεων των μεθόδων Runge-Kutta (RK1, RK2, RK3 & RK4) με σκοπό την αριθμητική επίλυση ενός ΠΑΤ με ΣΔΕ Bernoulli. Η συγκεκριμένη εξίσωση επιλέχθηκε επειδή παρέχει μια σαφή αναλυτική λύση, η οποία επιτρέπει την άμεση και ακριβή αξιολόγηση της ακρίβειας των αριθμητικών μεθόδων. Επομένως, μέσω αυτής της διαδικασίας, επιδιώκεται να διερευνηθεί η συμπεριφορά κάθε μεθόδου και να αναδειχθεί η αποτελεσματικότητά της σε σχέση με την αναλυτική λύση.

Η ΣΔΕ (με την αρχική της συνθήκη) που εξετάζεται έχει τη μορφή:

$$\frac{dy}{dt} = y^2e^t - y, y(0) = 0.5 \text{ (ΠΑΤ)}$$

και αντίστοιχη αναλυτική λύση:

$$y(t) = \frac{1}{e^t(2-t)}$$

Η επιλογή αυτής της εξίσωσης δεν είναι τυχαία. Με άλλα λόγια, η παρουσία μιας γνωστής αναλυτικής λύσης επιτρέπει τη σαφή σύγκριση των αριθμητικών αποτελεσμάτων με την πραγματική συμπεριφορά του συστήματος, γεγονός που καθιστά δυνατή την ακριβή μέτρηση των σφαλμάτων των διαφορετικών μεθόδων.

Επιβεβαίωση αναλυτικής λύσης:

Λύνουμε τη διαφορική εξίσωση:

$$\frac{dy}{dt} = y^2e^t - y$$

χρησιμοποιώντας την μεθοδολογία για επίλυση ΣΔΕ Bernoulli!

Βήμα 1: "Ταυτοποίηση ως εξίσωση Bernoulli"

Η εξίσωση Bernoulli έχει τη μορφή:

$$y' + p(t)y = q(t)y^r, r \in Z$$

Συγκρίνοντας με τη δοθείσα εξίσωση:

$$\frac{dy}{dt} + y = y^2e^t$$

Παρατηρούμε ότι: $p(t) = 1, q(t) = e^t$ & $r = 2$.

Βήμα 2: "Αλλαγή μεταβλητής"

Θέτουμε:

$$z = y^{1-2} = y^{-1}$$

και παραγωγίζουμε ως προς t:

$$\frac{dz}{dt} = -y^{-2}\frac{dy}{dt}$$

Ύστερα, αντικαθιστούμε από τη δοθείσα εξίσωση:

$$\frac{dz}{dt} = -y^{-2}(y^2e^t - y)$$

$$\frac{dz}{dt} = -(e^t - y^{-1})$$

$$\frac{dz}{dt} = -e^t + z$$

Βήμα 3: "Επίλυση της γραμμικής εξίσωσης για z"

Η εξίσωση που προέκυψε:

$$\frac{dz}{dt} - z = -e^t$$

είναι γραμμική διότι έχει γενική μορφή:

$$\frac{dz}{dt} = -p(t)z + q(t)$$

με: $p(t) = -1$ & $q(t) = -e^t$.

Βρίσκουμε τον ολοκληρωτικό παράγοντα:

$$\mu(t) = e^{\int -1\, dt} = e^{-t}$$

Έπειτα, πολλαπλασιάζουμε με τον ολοκληρωτικό παράγοντα:

$$e^{-t}\frac{dz}{dt} - e^{-t}z = -e^{-t}e^t$$

$$(e^{-t}z)' = -1$$

Ολοκληρώνουμε ως προς t:

$$e^{-t}z = -t + C$$

και λύνουμε ως προς z:

$$z = e^t(C - t)$$

Βήμα 4: "Επιστροφή στη μεταβλητή y"

Επειδή:

$$z = y^{-1}$$

έχουμε:

$$y^{-1} = e^t(C - t)$$

$$y = \frac{1}{e^t(C - t)}$$

όπου αυτή είναι η **γενική λύση**!

Βήμα 5: "Εφαρμογή της αρχικής συνθήκης"

Η αρχική συνθήκη είναι $y(0) = 0.5$, άρα:

$$0.5 = \frac{1}{e^0(C - 0)}$$

$$0.5 = \frac{1}{C}$$

$$C = 2$$

Άρα, η **ειδική λύση** είναι:

$$y = \frac{1}{e^t(2 - t)}$$

Ακολουθεί μια ενιαία επεξήγηση του κώδικα:

Ο κώδικας έχει σχεδιαστεί για να επιλύει αριθμητικά μια διαφορική εξίσωση πρώτης τάξης $\frac{dy}{dt}$ χρησιμοποιώντας τις μεθόδους Runge-Kutta διαφόρων τάξεων (RK1 έως RK4) και να συγκρίνει τα αποτελέσματα με την αναλυτική λύση. Αρχικά, εισάγονται οι βιβλιοθήκες *NumPy* για αριθμητικούς υπολογισμούς και *Matplotlib* για οπτικοποίηση. Η διαφορική εξίσωση ορίζεται στη συνάρτηση f(t, y), με την αναλυτική της λύση να δίνεται από τη συνάρτηση exact(t).

Οι μέθοδοι Runge-Kutta (RK1, RK2, RK3 & RK4) υλοποιούνται με βάση τον αριθμό των βημάτων και τα βάρη που χρησιμοποιούν για την προσέγγιση της λύσης. Η *RK1 (Euler)* βασίζεται σε ένα βήμα με την αρχική κλίση, η *RK2 (Midpoint)* υπολογίζει μια ενδιάμεση κλίση, η *RK3* συνδυάζει τρεις κλίσεις για μεγαλύτερη ακρίβεια, και η *RK4* χρησιμοποιεί τέσσερα βήματα, προσφέροντας την υψηλότερη αξιοπιστία.

Το πρόβλημα επιλύεται στο χρονικό διάστημα **[0, 1.9]**, με βήμα **h = 0.1**, και αρχική συνθήκη **y(0) = 0.5**. Συγκεκριμένα, επιλέχθηκε το άνω όριο να είναι το **t = 1.9** για να αποφευχθεί η ιδιομορφία (singularity) στο **t = 2**, όπου ο παρονομαστής της αναλυτικής λύσης μηδενίζεται. Για κάθε μέθοδο, οι προσεγγιστικές τιμές υπολογίζονται επαναληπτικά και αποθηκεύονται σε πίνακες.

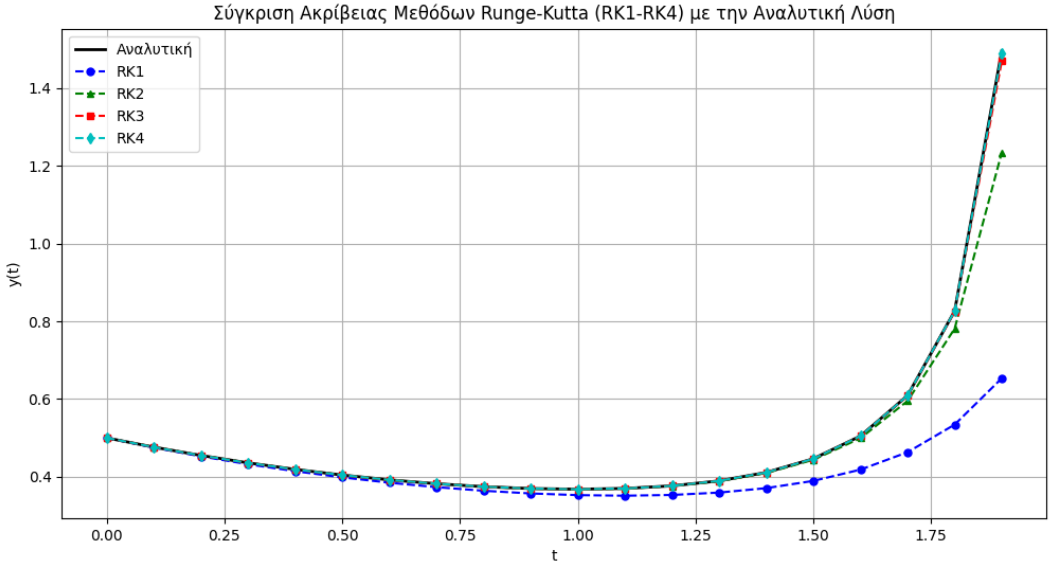
Στη συνέχεια, οι λύσεις των μεθόδων και η αναλυτική λύση απεικονίζονται σε κοινό γράφημα με διαφορετικά χρώματα για κάθε μέθοδο, επιτρέποντας τη γραφική σύγκριση της συμπεριφοράς τους. Για την ποσοτική ανάλυση, υπολογίζονται τα απόλυτα σφάλματα (|αριθμητική λύση - αναλυτική λύση|) για κάθε μέθοδο. Τα σφάλματα αυτά απεικονίζονται σε δύο γραφήματα:

- Το πρώτο χρησιμοποιεί γραμμική κλίμακα για να δείξει το μέγεθος των σφαλμάτων σε απόλυτες τιμές.
- Το δεύτερο χρησιμοποιεί λογαριθμική κλίμακα στον άξονα y, ώστε να αναδεικνύονται οι διαφορές στις μικρές τιμές σφαλμάτων, οι οποίες σε γραμμική κλίμακα θα ήταν δυσδιάκριτες.

Η διπλή απεικόνιση των σφαλμάτων επιτρέπει την ταυτόχρονη παρακολούθηση τόσο της τάξης μεγέθους των σφαλμάτων (μέσω της λογαριθμικής κλίμακας) όσο και της χρονικής εξέλιξής τους (μέσω της κανονικής κλίμακας). Αυτό υπογραμμίζει την επίδραση της τάξης κάθε μεθόδου (π.χ., RK4 έχει μικρότερα σφάλματα από RK1) και τη συσσώρευση σφαλμάτων καθώς το t πλησιάζει την ιδιομορφία (t → 2).

***Σημείωση*: Ο κώδικας βρίσκεται στο Παράρτημα Κώδικα!**

Ακολουθεί επεξήγηση των γραφημάτων του κώδικα:



Σχήμα 5.1: Σύγκριση Ακρίβειας Μεθόδων Runge-Kutta (RK1-RK4) με την Αναλυτική Λύση

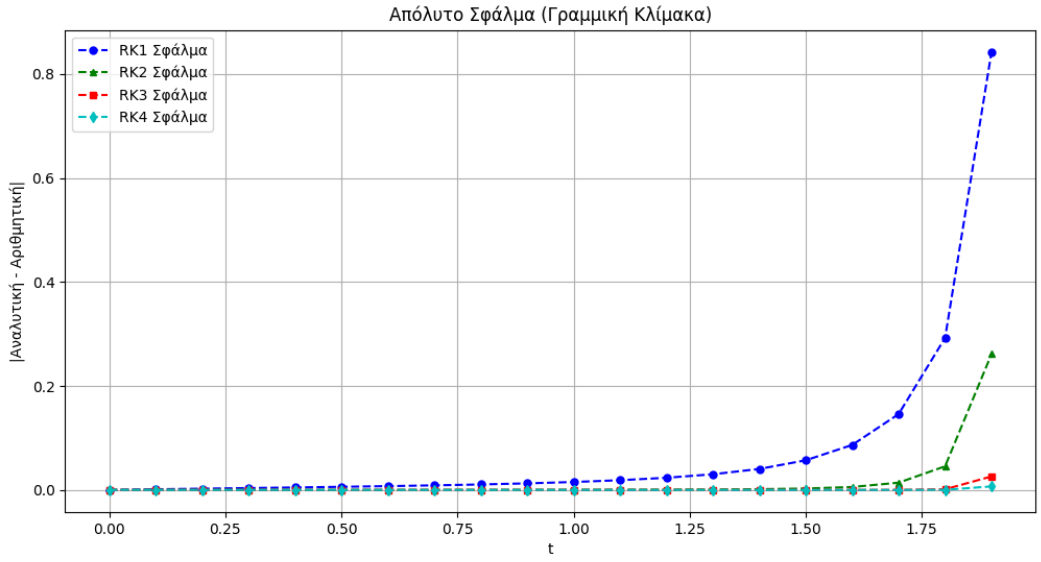
Το γράφημα με τίτλο "*Σύγκριση Ακρίβειας Μεθόδων Runge-Kutta (RK1-RK4) με την Αναλυτική Λύση*" απεικονίζει την αποδοτικότητα των αριθμητικών μεθόδων σε σύγκριση με την ακριβή λύση. Η αναλυτική λύση εμφανίζεται ως μια ομαλή, συνεχής καμπύλη (μαύρη γραμμή), αποτελώντας το σημείο αναφοράς. Οι μέθοδοι Runge-Kutta αναπαρίστανται με διαφορετικά χρώματα ή στυλ γραμμών, δείχνοντας διαφορετικά επίπεδα προσέγγισης.

Η **RK1 (Μέθοδος Euler)** παρουσιάζει τη μεγαλύτερη απόκλιση, ειδικά σε μεγαλύτερες τιμές του χρόνου, αποδεικνύοντας τη χαμηλή της ακρίβεια. Οι **RK2** και **RK3** εμφανίζουν σημαντική βελτίωση, αν και παραμένουν κάποιες αποκλίσεις σε ορισμένα σημεία. Αντίθετα, η **RK4** προσφέρει τη μεγαλύτερη ακρίβεια,

ακολουθώντας σχεδόν τέλεια την αναλυτική λύση ακόμη και για μεγάλα χρονικά διαστήματα.

Στα αρχικά στάδια της επίλυσης (t από 0.0 έως 0.5), όλες οι μέθοδοι φαίνονται να αποδίδουν ικανοποιητικά. Ωστόσο, με την αύξηση του χρόνου (t > 0.5 έως 1.9), η RK1 χάνει γρήγορα την ακρίβεια, ενώ η RK4 παραμένει εξαιρετικά σταθερή. Ιδιαίτερα σε σημεία με απότομες μεταβολές, οι μέθοδοι χαμηλότερης τάξης εμφανίζουν αστάθεια, σε αντίθεση με την RK4 που διατηρεί την ομαλότητα της λύσης.

Από πρακτική σκοπιά, η RK1 και η RK2 μπορούν να χρησιμοποιηθούν για ταχύτερους υπολογισμούς με αποδεκτή απώλεια ακρίβειας, ενώ σε εφαρμογές όπου η ακρίβεια είναι κρίσιμη, η RK4 αποτελεί την ιδανική επιλογή, παρά τον αυξημένο υπολογιστικό χρόνο. Το γράφημα καταδεικνύει ξεκάθαρα πως όσο υψηλότερη είναι η τάξη της μεθόδου Runge-Kutta, τόσο πιο αξιόπιστη και ακριβής είναι η προσέγγιση της πραγματικής λύσης, με την RK4 να ξεχωρίζει σε προβλήματα με πολύπλοκη δυναμική συμπεριφορά.



Σχήμα 5.2: Απόλυτο Σφάλμα (Γραμμική Κλίμακα)

Το γράφημα με τίτλο "*Απόλυτο Σφάλμα (Γραμμική Κλίμακα)*" απεικονίζει τη διαφορά μεταξύ των αριθμητικών μεθόδων Runge-Kutta (RK1-RK4) και της αναλυτικής λύσης σε διάφορες χρονικές στιγμές. Αρχικά, όλες οι μέθοδοι ξεκινούν με μηδενικό σφάλμα στο t = 0.0, καθώς μοιράζονται την ίδια αρχική συνθήκη. Ωστόσο, με την αύξηση του χρόνου, το σφάλμα αυξάνεται με διαφορετικούς ρυθμούς για κάθε μέθοδο.

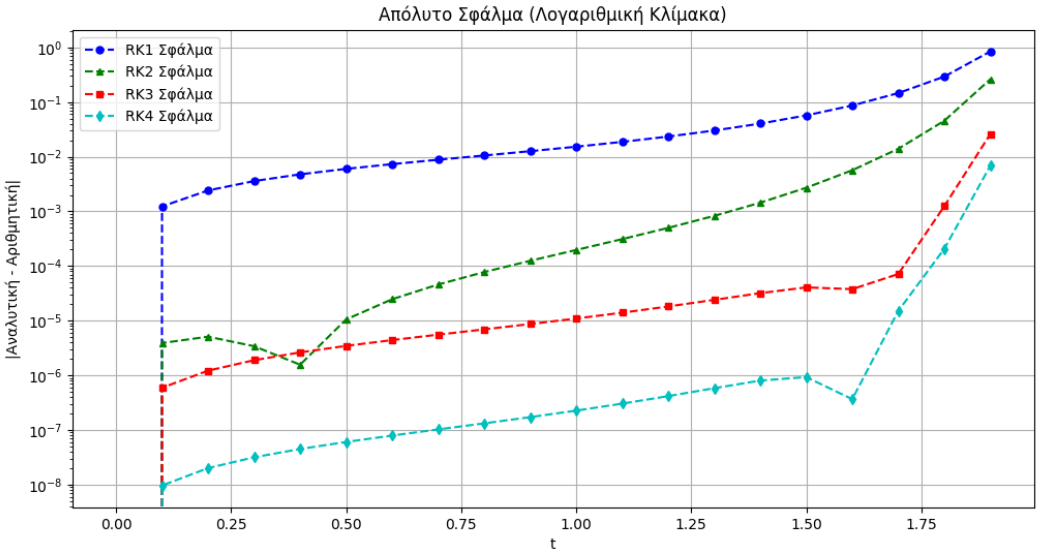
Η **RK1 (Μέθοδος Euler)** εμφανίζει τη μεγαλύτερη απόκλιση, με το σφάλμα να αυξάνεται ραγδαία, φτάνοντας σε τιμές άνω του 0.8 στο t = 1.9. Η **RK2** παρουσιάζει βελτιωμένη απόδοση, αν και το σφάλμα της παραμένει υψηλό για μεγάλα χρονικά διαστήματα. Η **RK3** προσφέρει ακόμα μεγαλύτερη ακρίβεια, παραμένοντας κοντά

στην **RK4** μέχρι το τέλος του διαστήματος. Η **RK4** αποδεικνύεται η πιο αξιόπιστη μέθοδος, με το σφάλμα της να παραμένει πολύ κοντά στο μηδέν σε όλη τη διάρκεια.

Σε κρίσιμα σημεία, όπως το διάστημα από t = 0.6 έως 1.0, το σφάλμα της **RK1** αυξάνεται αισθητά, ενώ η **RK2** ακολουθεί με μικρότερη απόκλιση. Οι μέθοδοι ανώτερης τάξης (**RK3** και **RK4**) διατηρούν σταθερότητα, παρουσιάζοντας ελάχιστες αποκλίσεις. Στο τέλος του χρονικού διαστήματος (t = 1.9), οι διαφορές μεταξύ των μεθόδων είναι εμφανείς, με τη **RK1** να παρουσιάζει πολύ μεγαλύτερο σφάλμα σε σχέση με τη **RK4**.

Από πρακτική σκοπιά, η **RK4** αποτελεί την ιδανική επιλογή για προβλήματα που απαιτούν υψηλή ακρίβεια, ακόμη και για μεγάλα χρονικά διαστήματα. Αντίθετα, η **RK1** είναι κατάλληλη μόνο για πολύ μικρές χρονικές περιόδους ή όταν η ταχύτητα υπολογισμών υπερισχύει της ακρίβειας.

Συνολικά, το γράφημα επιβεβαιώνει ότι όσο υψηλότερη είναι η τάξη της μεθόδου Runge-Kutta, τόσο μικρότερο είναι το απόλυτο σφάλμα σε σύγκριση με την αναλυτική λύση. Η **RK4** διατηρεί υψηλή ακρίβεια ακόμα και για μεγάλα χρονικά διαστήματα, ενώ η **RK1** χάνει γρήγορα την αξιοπιστία της.



Σχήμα 5.3: Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)

Το γράφημα με τίτλο "*Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)*" παρουσιάζει την ακρίβεια των μεθόδων Runge-Kutta (RK1-RK4) σε σχέση με την αναλυτική λύση, απεικονίζοντας το απόλυτο σφάλμα σε λογαριθμική κλίμακα. Στον οριζόντιο άξονα αποτυπώνεται ο χρόνος t από 0.0 έως 1.9, ενώ στον κάθετο άξονα εμφανίζεται ο λογάριθμος του σφάλματος, επιτρέποντας την ευκρινή παρακολούθηση ακόμη και των μικρότερων αποκλίσεων.

Η συμπεριφορά του σφάλματος διαφοροποιείται έντονα ανάλογα με την τάξη της μεθόδου. Η **RK1 (Μέθοδος Euler)** παρουσιάζει την ταχύτερη αύξηση

σφάλματος, με την καμπύλη της να έχει απότομη κλίση, δείχνοντας πως το σφάλμα αυξάνεται εκθετικά καθώς ο χρόνος προχωρά. Οι **RK2** και **RK3** επιδεικνύουν σαφή βελτίωση, με την **RK3** να υπερέχει εμφανώς έναντι της **RK2**, καθώς το σφάλμα αυξάνεται με πιο αργό ρυθμό και η καμπύλη της παραμένει πιο "επίπεδη". Η **RK4**, ωστόσο, ξεχωρίζει, με την καμπύλη του σφάλματός της να παραμένει σχεδόν οριζόντια, γεγονός που υποδεικνύει εξαιρετική ακρίβεια και σταθερότητα καθ' όλη τη διάρκεια του χρονικού διαστήματος.

Στα αρχικά στάδια (t από 0.0 έως 0.5), όλες οι μέθοδοι διατηρούν πολύ μικρό σφάλμα. Ωστόσο, με την πάροδο του χρόνου (t>0.6), οι διαφορές γίνονται εμφανείς. Το σφάλμα της RK1 αυξάνεται δραματικά, αποδεικνύοντας πως αυτή η μέθοδος δεν είναι κατάλληλη για μακροπρόθεσμες προσομοιώσεις. Αντίθετα, η **RK4** συνεχίζει να παρουσιάζει εξαιρετική απόδοση, διατηρώντας το σφάλμα σε σταθερά χαμηλά επίπεδα, ακόμα και σε περιπτώσεις όπου το πρόβλημα ενδέχεται να εμφανίζει αυξημένη πολυπλοκότητα ή απότομες μεταβολές.

Μια αξιοσημείωτη παρατήρηση στο γράφημα αφορά τη σύγκριση μεταξύ των μεθόδων **RK2** και **RK3**. Παρόλο που η **RK3** είναι θεωρητικά ανώτερη, παρατηρείται ότι γύρω από το t=0.4, η **RK2** εμφανίζει μικρότερο σφάλμα από την RK3. Αυτό μπορεί να οφείλεται σε διάφορους παράγοντες, όπως η επιλογή ακατάλληλου βήματος h, το οποίο μπορεί να είναι πολύ μεγάλο για τη **RK3**, περιορίζοντας την ικανότητά της να αποδώσει τα αναμενόμενα αποτελέσματα. Επιπλέον, η ίδια η φύση της διαφορικής εξίσωσης μπορεί να δημιουργεί τοπικές ιδιομορφίες ή απότομες μεταβολές που διαχειρίζεται καλύτερα η **RK2** λόγω διαφορετικών συντελεστών βάρους και ενδιάμεσων σημείων. Σε ορισμένες περιπτώσεις, ακόμα και σφάλματα στρογγυλοποίησης μπορεί να επηρεάσουν τα αποτελέσματα, ιδίως σε προβλήματα όπου το συνολικό σφάλμα είναι ήδη πολύ μικρό.

Το γράφημα αναδεικνύει ξεκάθαρα τη σύνδεση μεταξύ της τάξης της μεθόδου Runge-Kutta και της ακρίβειας της προσέγγισης. Οι χαμηλότερης τάξης μέθοδοι (**RK1** και **RK2**) εμφανίζουν ταχύτερη αύξηση του σφάλματος, καθιστώντας τις κατάλληλες μόνο για προβλήματα μικρής χρονικής κλίμακας ή για περιπτώσεις όπου η ακρίβεια δεν είναι κρίσιμη. Αντίθετα, η **RK4** αποδεικνύεται η ιδανική επιλογή για προβλήματα που απαιτούν υψηλή ακρίβεια και μακροπρόθεσμη σταθερότητα, παρά το μεγαλύτερο υπολογιστικό κόστος που συνεπάγεται η εφαρμογή της.

Η λογαριθμική κλίμακα καθιστά ιδιαίτερα εμφανείς τις διαφορές στην απόδοση μεταξύ των μεθόδων, ειδικά σε μεγαλύτερα χρονικά διαστήματα. Το τελικό συμπέρασμα που προκύπτει είναι σαφές: όσο υψηλότερη είναι η τάξη της μεθόδου Runge-Kutta, τόσο μεγαλύτερη είναι η αξιοπιστία και η ακρίβεια της προσέγγισης στην επίλυση διαφορικών εξισώσεων, με την RK4 να ξεχωρίζει ως η πλέον αξιόπιστη επιλογή για απαιτητικά προβλήματα με πολύπλοκη δυναμική συμπεριφορά.

5.2 Υλοποίηση ενός Νευρωνικού Δικτύου για την προσέγγιση λύσης ΣΔΕ (με τη χρήση του αριθμητικού αλγορίθμου Runge-Kutta 4ου βαθμού (RK4) ως προσαρμοσμένου βελτιστοποιητή)

Σε αυτήν την ενότητα, παρουσιάζεται η υλοποίηση ενός νευρωνικού δικτύου που χρησιμοποιεί τον αλγόριθμο **Runge-Kutta 4ης τάξης (RK4)** ως προσαρμοσμένο βελτιστοποιητή για την προσέγγιση της λύσης του ΠΑΤ της παρακάτω ΣΔΕ:

$$\frac{dy}{dt} = y^2 e^t - y, y(0) = 0.5$$

με αντίστοιχη αναλυτική λύση:

$$y(t) = \frac{1}{e^t(2-t)}$$

Ακολουθεί μια ενιαία επεξήγηση του κώδικα:

Ο κώδικας έχει σχεδιαστεί για να επιλύει αριθμητικά μια διαφορική εξίσωση πρώτης τάξης μέσω της εκπαίδευσης ενός νευρωνικού δικτύου χρησιμοποιώντας μια προσαρμοσμένη υλοποίηση της μεθόδου Runge-Kutta 4ου βαθμού (RK4) ως βελτιστοποιητή. Αρχικά, εισάγονται οι βιβλιοθήκες PyTorch για την υλοποίηση και εκπαίδευση του δικτύου, NumPy για τη δημιουργία των δεδομένων εκπαίδευσης και Matplotlib για την απεικόνιση των αποτελεσμάτων.

Το νευρωνικό δίκτυο ορίζεται ως μια κλάση SimpleNN, η οποία υλοποιεί ένα πλήρως συνδεδεμένο δίκτυο (MLP). Το δίκτυο δέχεται μία τιμή εισόδου, επεξεργάζεται την πληροφορία μέσω ενός κρυφού επιπέδου με 10 νευρώνες και ενεργοποίηση ReLU, και εξάγει μία τιμή, η οποία αντιπροσωπεύει την προσέγγιση της λύσης της διαφορικής εξίσωσης.

Η συνάρτηση κόστους που χρησιμοποιείται είναι το μέσο τετραγωνικό σφάλμα (MSE), το οποίο μετρά την απόκλιση των προβλέψεων του δικτύου από την αναλυτική λύση της εξίσωσης. Το MSE υπολογίζεται λαμβάνοντας το τετραγωνικό σφάλμα κάθε πρόβλεψης και στη συνέχεια το μέσο όρο όλων των σφαλμάτων.

Η μέθοδος Runge-Kutta 4ου βαθμού (RK4) ενσωματώνεται στον βελτιστοποιητή για την ενημέρωση των παραμέτρων του δικτύου. Αντί να χρησιμοποιείται μια κλασική μέθοδος όπως το Gradient Descent, ο RK4 υπολογίζει τέσσερις διαφορετικές εκτιμήσεις της κλίσης (k1, k2, k3, k4) για να πετύχει ακριβέστερη ενημέρωση των παραμέτρων. Οι εκτιμήσεις αυτές υπολογίζονται με βάση το τρέχον σύνολο παραμέτρων και προσωρινές ενημερώσεις, διασφαλίζοντας ότι δεν επηρεάζονται οι πραγματικές παράμετροι του μοντέλου μέχρι την τελική συνδυαστική ενημέρωση.

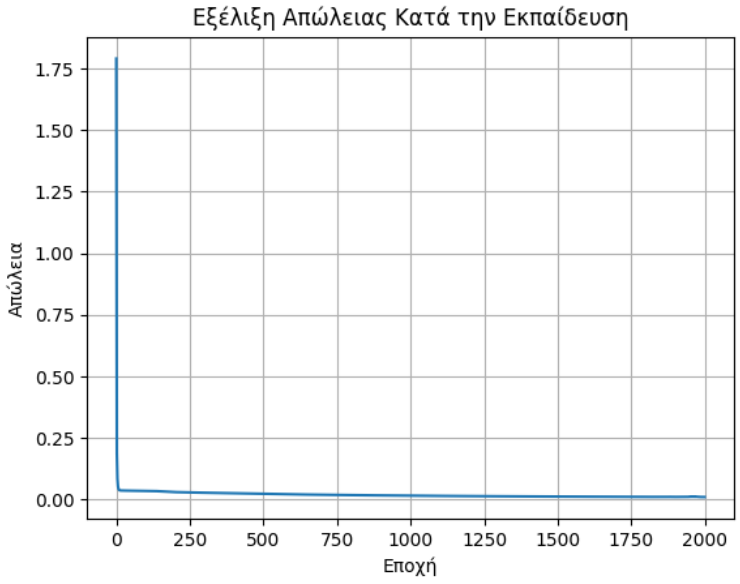
Τα δεδομένα εκπαίδευσης παράγονται από την αναλυτική λύση της διαφορικής εξίσωσης και η εκπαίδευση πραγματοποιείται στο διάστημα [0, 1.9], αποφεύγοντας το σημείο t=2, όπου η συνάρτηση απειρίζεται. Τα δεδομένα μετατρέπονται σε tensors του PyTorch, ώστε να είναι συμβατά με το νευρωνικό δίκτυο.

Η διαδικασία εκπαίδευσης εκτελείται για 2000 εποχές, εφαρμόζοντας σε κάθε επανάληψη τον RK4 βελτιστοποιητή για την ενημέρωση των βαρών του δικτύου. Η εξέλιξη της απώλειας παρακολουθείται και εμφανίζεται σε τακτά χρονικά διαστήματα, επιτρέποντας την αξιολόγηση της απόδοσης του μοντέλου.

Τέλος, τα αποτελέσματα απεικονίζονται μέσω γραφημάτων. Το πρώτο γράφημα παρουσιάζει τη μείωση του MSE κατά τη διάρκεια της εκπαίδευσης, ενώ το δεύτερο συγκρίνει την αναλυτική λύση με την πρόβλεψη του νευρωνικού δικτύου. Στο γράφημα αυτό, η αναλυτική λύση εμφανίζεται με συνεχή γραμμή και η πρόβλεψη του δικτύου με διακεκομμένη, προσφέροντας μια οπτική σύγκριση της ακρίβειας της προσέγγισης. Ο στόχος είναι να διαπιστωθεί αν το νευρωνικό δίκτυο μπορεί να προσεγγίσει επαρκώς την αναλυτική λύση μέσω της εκπαίδευσης με τον custom RK4 βελτιστοποιητή.

<u>Σημείωση</u>: Ο κώδικας βρίσκεται στο Παράρτημα Κώδικα!

Ακολουθεί επεξήγηση των γραφημάτων του κώδικα:

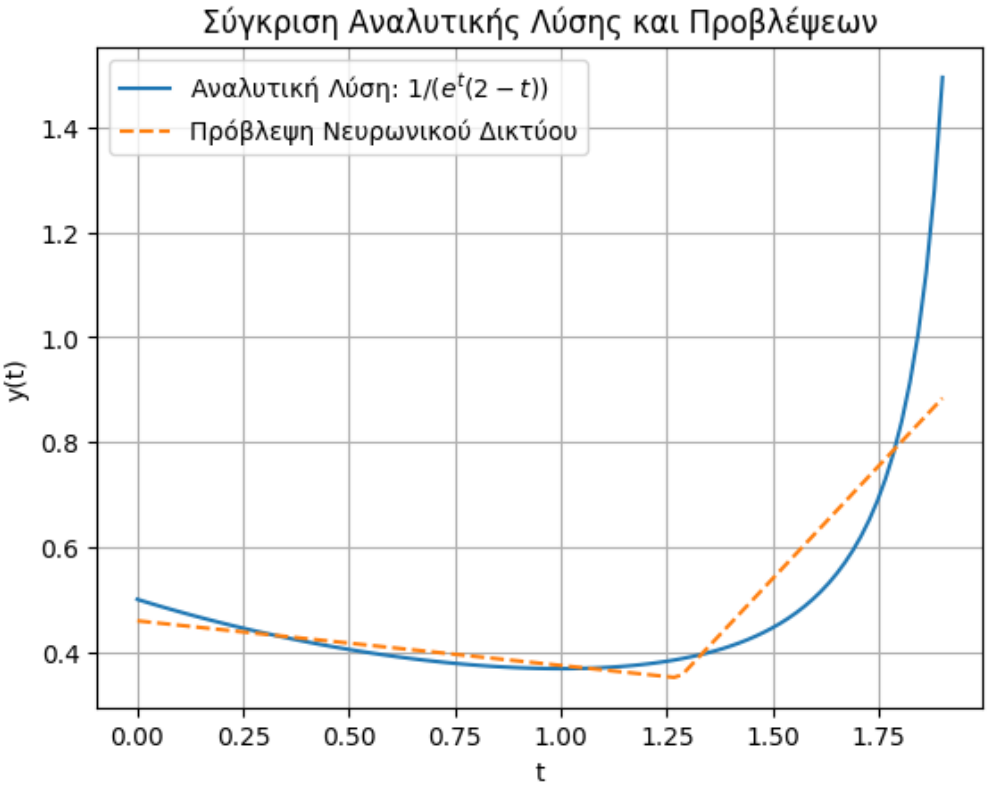


Σχήμα 5.4: Εξέλιξη Απώλειας Κατά την Εκπαίδευση (RK4)

Στο γράφημα "*Εξέλιξη Απώλειας Κατά την Εκπαίδευση*" φαίνεται η **πτώση της απώλειας Μέσου Τετραγωνικού Σφάλματος (MSE)** καθώς προχωρά η εκπαίδευση του μοντέλου. Αρχικά, η απώλεια ξεκινά από υψηλές τιμές και μειώνεται απότομα στις πρώτες 100 εποχές, φτάνοντας σε πολύ χαμηλά επίπεδα στο τέλος της διαδικασίας. Η καμπύλη εμφανίζει **σταθερή τάση μείωσης** χωρίς

μεγάλες διακυμάνσεις, γεγονός που υποδηλώνει ότι το μοντέλο μαθαίνει αποτελεσματικά και βελτιώνεται συνεχώς. Δεν παρατηρείται υπερεκπαίδευση (overfitting) ή σταθεροποίηση της απώλειας πριν το τέλος των εποχών, πράγμα που επιβεβαιώνει την επιτυχή σύγκλιση της εκπαίδευσης.

Συμπερασματικά, το μοντέλο **βελτιώνεται ομαλά** με την πάροδο του χρόνου, με το σφάλμα να μειώνεται σημαντικά από την αρχή μέχρι το τέλος της εκπαίδευσης. Αυτό δείχνει ότι η διαδικασία εκπαίδευσης είναι **αποδοτική** και ότι το νευρωνικό δίκτυο έχει την ικανότητα να προσεγγίζει την αναλυτική λύση με υψηλή ακρίβεια. Επίσης, το νευρωνικό δίκτυο μαθαίνει και βελτιώνεται σταδιακά, αλλά με μικρότερες βελτιώσεις όσο προχωρά η εκπαίδευση.



Σχήμα 5.5: Σύγκριση Αναλυτικής Λύσης και Προβλέψεων (RK4)

Στο γράφημα "*Σύγκριση Αναλυτικής Λύσης και Προβλέψεων*" συγκρίνονται η **αναλυτική λύση** (συνεχής μπλε καμπύλη) και οι **προβλέψεις του νευρωνικού δικτύου** (διακεκομμένη πορτοκαλί καμπύλη) για τη συνάρτηση y(t). Παρατηρείται ότι οι δύο καμπύλες **ταυτίζονται σε ικανοποιητικό βαθμό** στο μεγαλύτερο μέρος του εύρους του t, γεγονός που υποδηλώνει υψηλή ακρίβεια του μοντέλου. Σε μερικά σημεία, ειδικά καθώς το t πλησιάζει προς το τέλος του πεδίου ορισμού, η διακεκομμένη γραμμή εμφανίζει **ελαφρές αποκλίσεις** από την αναλυτική λύση, αλλά χωρίς να αλλοιώνει την γενική τάση.

Συμπερασματικά, το νευρωνικό δίκτυο **προσδιορίζει με επιτυχία** τη συμπεριφορά της συνάρτησης y(t), με προβλέψεις που ακολουθούν στενά την

πραγματική λύση. Οι μικρές αποκλίσεις σε υψηλές τιμές του t μπορούν να αποδοθούν σε περιορισμούς του μοντέλου ή στη δυσκολία προσέγγισης κοντά σε κρίσιμα σημεία (π.χ. όπου η συνάρτηση αλλάζει απότομα). Η συνολική απόδοση είναι **ικανοποιητική** και επιβεβαιώνει την αποτελεσματικότητα της εκπαίδευσης.

5.3 Υλοποίηση ενός Νευρωνικού Δικτύου για την Προσέγγιση Λύσης ΣΔΕ (με Full-Batch Gradient Descent ως βελτιστοποιητή)

Σε αυτήν την ενότητα, παρουσιάζεται η υλοποίηση ενός νευρωνικού δικτύου που χρησιμοποιεί τον αλγόριθμο **Full-Batch Gradient Descent**, ο οποίος βασίζεται στην ενημέρωση των βαρών χρησιμοποιώντας *όλο το dataset ανά εποχή,* ως βελτιστοποιητή για την προσέγγιση της λύσης του ΠΑΤ της παρακάτω ΣΔΕ:

$$\frac{dy}{dt} = y^2 e^t - y, y(0) = 0.5$$

με αντίστοιχη αναλυτική λύση:

$$y(t) = \frac{1}{e^t(2-t)}$$

Ακολουθεί μια ενιαία επεξήγηση του κώδικα:

Ο κώδικας επιλύει αριθμητικά μια διαφορική εξίσωση πρώτης τάξης εκπαιδεύοντας ένα νευρωνικό δίκτυο με τη μέθοδο Full-Batch Gradient Descent. Αρχικά, εισάγονται οι απαραίτητες βιβλιοθήκες: PyTorch για την υλοποίηση και εκπαίδευση του δικτύου, NumPy για τη δημιουργία των δεδομένων εκπαίδευσης και Matplotlib για την απεικόνιση των αποτελεσμάτων. Το νευρωνικό δίκτυο υλοποιείται μέσω της κλάσης SimpleNN, η οποία αποτελεί ένα πλήρως συνδεδεμένο πολυεπίπεδο δίκτυο. Το δίκτυο δέχεται μία τιμή εισόδου, την επεξεργάζεται μέσω ενός κρυφού επιπέδου με δέκα νευρώνες και λειτουργία ενεργοποίησης ReLU, και επιστρέφει μία τιμή εξόδου που στοχεύει στην προσέγγιση της λύσης της διαφορικής εξίσωσης.

Η συνάρτηση κόστους που χρησιμοποιείται είναι το μέσο τετραγωνικό σφάλμα (MSE), το οποίο μετρά την απόκλιση των προβλέψεων του δικτύου από την αναλυτική λύση, αξιολογώντας έτσι την απόδοση του μοντέλου. Για την εκπαίδευση εφαρμόζεται η μέθοδος Full-Batch Gradient Descent, η οποία ενημερώνει τα βάρη του δικτύου χρησιμοποιώντας ολόκληρο το σύνολο δεδομένων σε κάθε επανάληψη. Αυτή η μέθοδος είναι πιο σταθερή από το Stochastic Gradient Descent, αλλά μπορεί να είναι πιο αργή και συχνά απαιτεί μικρότερο ρυθμό μάθησης για να αποφευχθούν ταλαντώσεις.

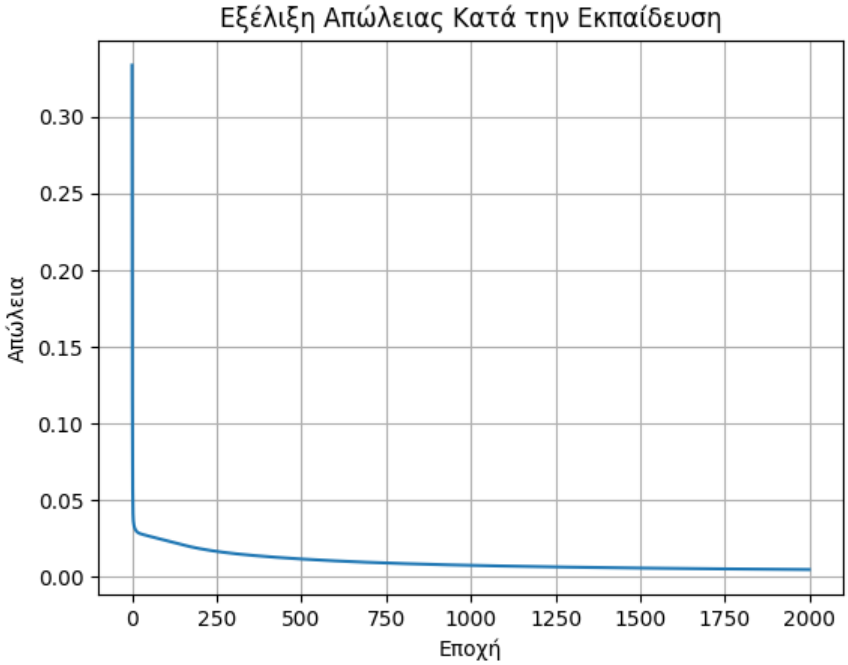
Τα δεδομένα εκπαίδευσης προκύπτουν από την αναλυτική λύση της διαφορικής εξίσωσης και δημιουργούνται στο διάστημα από 0 έως 1.9, αποφεύγοντας το σημείο όπου η συνάρτηση απειρίζεται. Στη συνέχεια, τα

δεδομένα μετατρέπονται σε μορφή κατάλληλη για χρήση από το δίκτυο, αξιοποιώντας τους PyTorch tensors. Η εκπαίδευση εκτελείται για 2000 εποχές με ρυθμό μάθησης 0.1. Σε κάθε εποχή πραγματοποιείται υπολογισμός των προβλέψεων, εκτίμηση της απώλειας, υπολογισμός των παραγώγων και ενημέρωση των βαρών με βάση τον κανόνα του gradient descent.

Η αξιολόγηση των αποτελεσμάτων γίνεται μέσω γραφημάτων που απεικονίζουν την πορεία της εκπαίδευσης. Το πρώτο γράφημα δείχνει την πτώση της τιμής της απώλειας σε βάθος χρόνου, ενώ το δεύτερο συγκρίνει την αναλυτική λύση με την πρόβλεψη του νευρωνικού δικτύου, προσφέροντας μια οπτική αξιολόγηση της ακρίβειας του μοντέλου. Ο κύριος στόχος είναι να επιβεβαιωθεί η ικανότητα του δικτύου να προσεγγίζει αποτελεσματικά τη θεωρητική λύση της διαφορικής εξίσωσης μέσω της εκπαίδευσης με Full-Batch Gradient Descent.

<u>Σημείωση</u>: Ο κώδικας βρίσκεται στο Παράρτημα Κώδικα!

Ακολουθεί επεξήγηση των γραφημάτων του κώδικα:

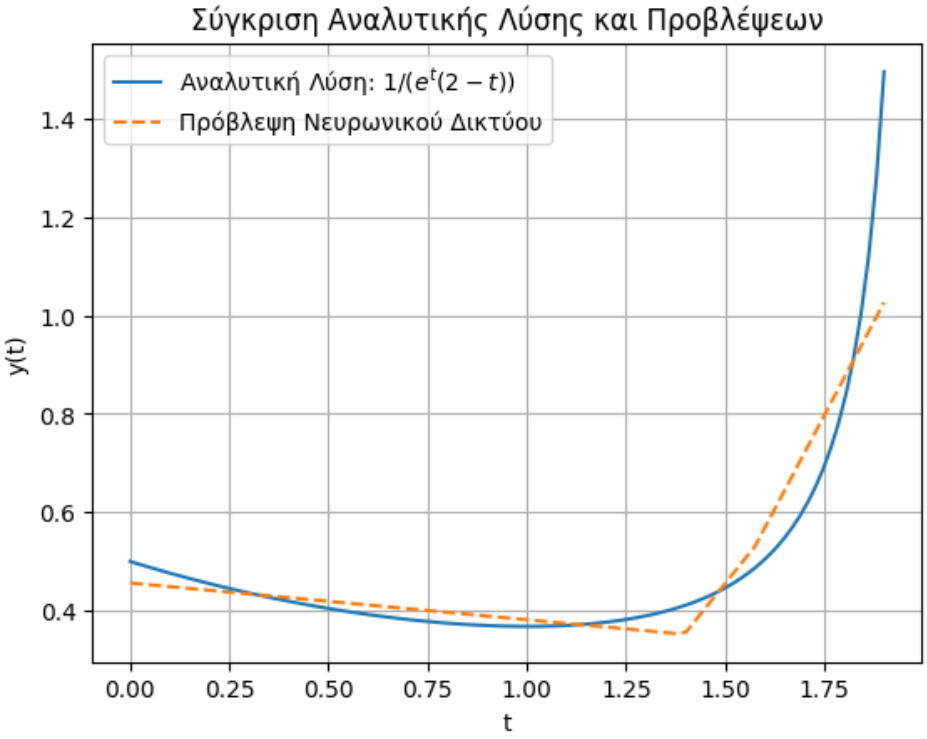


Σχήμα 5.6: Εξέλιξη Απώλειας Κατά την Εκπαίδευση (Full-Batch GD)

Το γράφημα **"*Εξέλιξη Απώλειας Κατά την Εκπαίδευση*"** δείχνει τη **μείωση της απώλειας** κατά τη διάρκεια της εκπαίδευσης. Στην αρχή, η απώλεια είναι πολύ υψηλή, αλλά μειώνεται απότομα στις πρώτες 100 εποχές. Στη συνέχεια, η πτώση συνεχίζεται με πιο αργό ρυθμό μέχρι να φτάσει σε πολύ χαμηλές τιμές.

Συμπερασματικά, το νευρωνικό δίκτυο **μαθαίνει γρήγορα στα πρώτα στάδια**, μειώνοντας απότομα την απώλεια. Καθώς προχωρά η εκπαίδευση, οι αλλαγές γίνονται πιο μικρές, δείχνοντας ότι το μοντέλο πλησιάζει τη βέλτιστη λύση. Το τελικό χαμηλό επίπεδο απώλειας επιβεβαιώνει ότι το δίκτυο έχει **μάθει σωστά**

τη λύση, ενώ η ομαλή μείωση της καμπύλης δείχνει μια **σταθερή και αποτελεσματική διαδικασία εκπαίδευσης**.



Σχήμα 5.7: Σύγκριση Αναλυτικής Λύσης και Προβλέψεων (Full-Batch GD)

Στο γράφημα "*Σύγκριση Αναλυτικής Λύσης και Προβλέψεων*" συγκρίνονται η **πραγματική λύση** της διαφορικής εξίσωσης (μπλε συνεχής γραμμή) και η **πρόβλεψη του νευρωνικού δικτύου** (πορτοκαλί διακεκομμένη γραμμή). Παρατηρείται πολύ καλή συμφωνία μεταξύ των δύο καμπυλών στο μεγαλύτερο μέρος του διαστήματος, ειδικά για μικρές τιμές του t. Στις μεγαλύτερες τιμές του t, η πρόβλεψη του δικτύου αποκλίνει ελαφρώς από την αναλυτική λύση.

Συμπερασματικά, το νευρωνικό δίκτυο καταφέρνει **να προσεγγίσει με ακρίβεια** τη λύση της διαφορικής εξίσωσης, ειδικά στο αρχικό και μεσαίο τμήμα του διαστήματος. Η μικρή απόκλιση προς το τέλος πιθανώς οφείλεται στη δυσκολία εκμάθησης των **απότομων αλλαγών** της συνάρτησης ή στη συμπεριφορά του Full-Batch GD. Παρόλα αυτά, η συνολική προσέγγιση είναι **ιδιαίτερα ικανοποιητική**, επιβεβαιώνοντας ότι το μοντέλο έχει μάθει τη γενική μορφή της λύσης.

5.4 Υλοποίηση ενός Νευρωνικού Δικτύου για την Προσέγγιση Λύσης ΣΔΕ (με τον βελτιστοποιητή Adam)

Σε αυτήν την ενότητα, παρουσιάζεται η υλοποίηση ενός νευρωνικού δικτύου που χρησιμοποιεί τον αλγόριθμο Adam, ο οποίος συνδυάζει τα πλεονεκτήματα

των **RMSProp** και **Momentum** για γρήγορη και σταθερή σύγκλιση, ως βελτιστοποιητή για την προσέγγιση της λύσης του ΠΑΤ της παρακάτω ΣΔΕ:

$$\frac{dy}{dt} = y^2 e^t - y, y(0) = 0.5$$

με αντίστοιχη αναλυτική λύση:

$$y(t) = \frac{1}{e^t(2-t)}$$

Ακολουθεί μια ενιαία επεξήγηση του κώδικα:

Ο κώδικας αυτός επιλύει αριθμητικά μια διαφορική εξίσωση πρώτης τάξης μέσω της εκπαίδευσης ενός νευρωνικού δικτύου με τη μέθοδο βελτιστοποίησης Adam. Αρχικά, εισάγονται οι απαραίτητες βιβλιοθήκες: η PyTorch χρησιμοποιείται για τη δημιουργία και εκπαίδευση του νευρωνικού δικτύου, η NumPy για τη δημιουργία των δεδομένων εκπαίδευσης, και η Matplotlib για την οπτικοποίηση των αποτελεσμάτων. Το νευρωνικό δίκτυο υλοποιείται μέσω της κλάσης SimpleNN, η οποία είναι ένα πλήρως συνδεδεμένο πολυεπίπεδο δίκτυο. Το δίκτυο δέχεται ως είσοδο μία τιμή, την επεξεργάζεται μέσω ενός κρυφού επιπέδου που περιλαμβάνει δέκα νευρώνες με λειτουργία ενεργοποίησης ReLU και αποδίδει μία τιμή εξόδου που προσεγγίζει τη λύση της διαφορικής εξίσωσης.

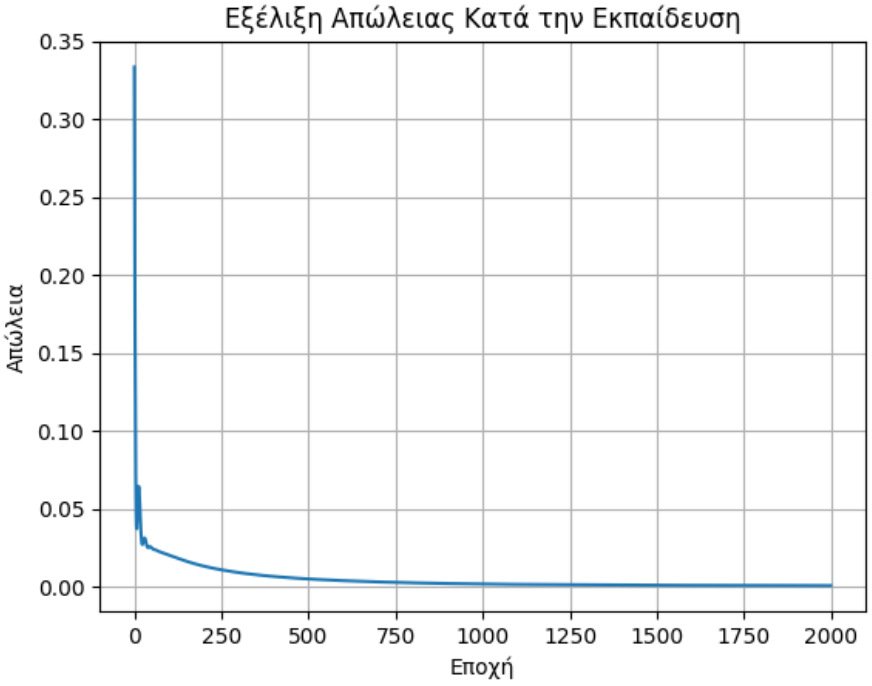
Η συνάρτηση κόστους που χρησιμοποιείται είναι το μέσο τετραγωνικό σφάλμα (MSE), το οποίο μετρά την απόκλιση των προβλέψεων του δικτύου από την αναλυτική λύση, προσφέροντας έναν σαφή δείκτη της απόδοσης του μοντέλου. Η εκπαίδευση βασίζεται στον βελτιστοποιητή Adam, ο οποίος συνδυάζει τα πλεονεκτήματα του Momentum και της προσαρμογής του ρυθμού μάθησης για κάθε παράμετρο. Ο Adam επιταχύνει τη σύγκλιση και επιλύει προβλήματα που προκύπτουν από μεταβαλλόμενους ρυθμούς μάθησης, καθιστώντας τον ιδιαίτερα αποδοτικό για την εκπαίδευση νευρωνικών δικτύων. Στο PyTorch, η υλοποίησή του γίνεται εύκολα μέσω της κλάσης torch.optim.Adam.

Τα δεδομένα εκπαίδευσης προέρχονται από την αναλυτική λύση της διαφορικής εξίσωσης. Αρχικά, δημιουργούνται 100 σημεία στο διάστημα [0, 1.9], αποφεύγοντας το σημείο t=2, όπου η συνάρτηση απειρίζεται. Τα δεδομένα μετατρέπονται σε PyTorch tensors, καθιστώντας τα κατάλληλα για χρήση στην εκπαίδευση. Η διαδικασία εκπαίδευσης πραγματοποιείται για 2000 εποχές με ρυθμό μάθησης 0.01. Σε κάθε εποχή εκτελείται ένας πλήρης κύκλος εκπαίδευσης: το δίκτυο υπολογίζει τις προβλέψεις (forward pass), εκτιμά την απώλεια συγκρίνοντας τις προβλέψεις με την αναλυτική λύση, υπολογίζει τα gradients της απώλειας ως προς τα βάρη (backward pass) και, τέλος, προσαρμόζει τα βάρη με βάση τα gradients και τους ρυθμούς μάθησης που προσαρμόζει αυτόματα ο Adam.

Η αξιολόγηση των αποτελεσμάτων πραγματοποιείται μέσω γραφημάτων. Το πρώτο γράφημα απεικονίζει τη μείωση της απώλειας (MSE) κατά τη διάρκεια της εκπαίδευσης, δίνοντας μια σαφή εικόνα της προόδου του δικτύου. Το δεύτερο γράφημα συγκρίνει την αναλυτική λύση με τις προβλέψεις του νευρωνικού δικτύου: η θεωρητική λύση παρουσιάζεται με συνεχή γραμμή, ενώ οι προβλέψεις του μοντέλου αποδίδονται με διακεκομμένη γραμμή. Ο στόχος είναι να διαπιστωθεί αν το νευρωνικό δίκτυο μπορεί να μάθει και να προσεγγίσει ικανοποιητικά την αναλυτική λύση, αξιοποιώντας τον βελτιστοποιητή Adam για την εκπαίδευση.

<u>Σημείωση</u>: Ο κώδικας βρίσκεται στο Παράρτημα Κώδικα!

Ακολουθεί επεξήγηση των γραφημάτων του κώδικα:

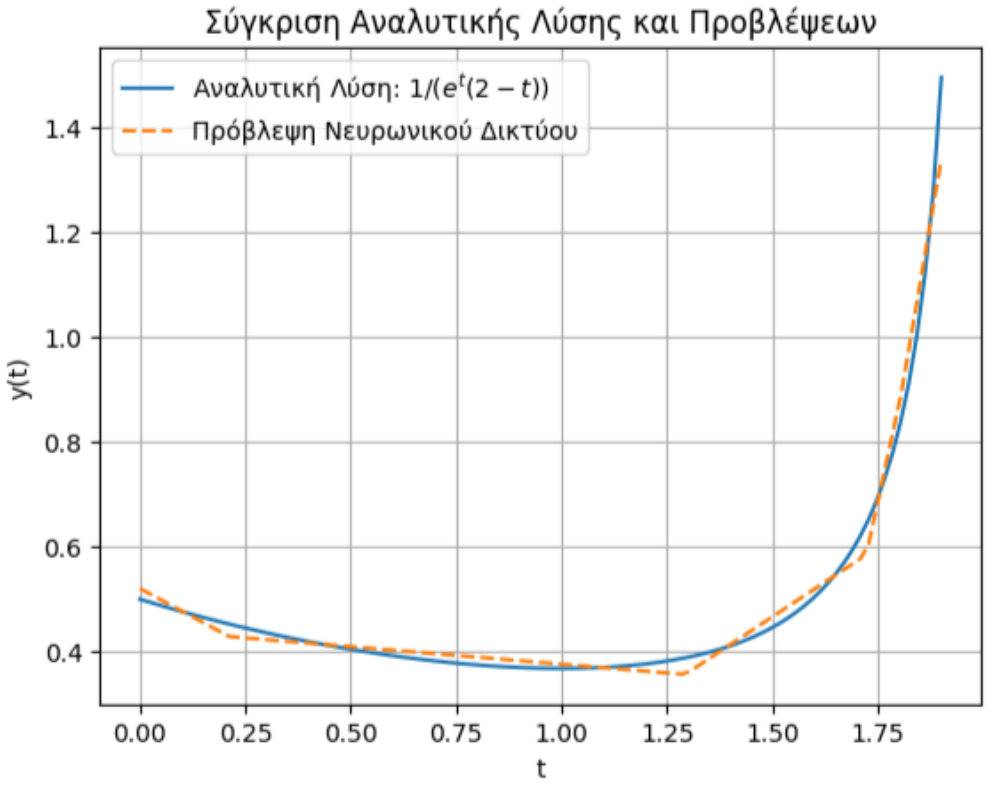


Σχήμα 5.8: Εξέλιξη Απώλειας Κατά την Εκπαίδευση (Adam)

Στο γράφημα "*Εξέλιξη Απώλειας Κατά την Εκπαίδευση*" φαίνεται η **σταθερή μείωση της απώλειας** καθώς αυξάνονται οι εποχές εκπαίδευσης. Η καμπύλη ξεκινά από υψηλές τιμές και πέφτει γρήγορα τις πρώτες 100 εποχές, ενώ στη συνέχεια συνεχίζει να μειώνεται με πιο αργό ρυθμό μέχρι να φτάσει σχεδόν στο μηδέν. Δεν εμφανίζονται απότομες ανόδους ή «πτώσεις» στην απώλεια, κάτι που δείχνει **σταθερή και ομαλή εκπαίδευση** χωρίς αστάθεια.

Συμπερασματικά, το μοντέλο **μαθαίνει αποτελεσματικά**, με την απώλεια να μειώνεται συνεχώς καθώς περνούν οι εποχές. Η πτώση της απώλειας στο τέλος των 2000 εποχών δείχνει ότι το νευρωνικό δίκτυο έχει **προσαρμοστεί καλά** στα δεδομένα και επιτυγχάνει υψηλή ακρίβεια. Η απουσία διακυμάνσεων

επιβεβαιώνει ότι η διαδικασία εκπαίδευσης ήταν **ελεγχόμενη και βελτιστοποιημένη**.



Σχήμα 5.9: Σύγκριση Αναλυτικής Λύσης και Προβλέψεων (Adam)

Στο γράφημα "*Σύγκριση Αναλυτικής Λύσης και Προσομοίωσης*" φαίνεται η **σύγκριση μεταξύ της αναλυτικής λύσης** (μπλε συνεχής καμπύλη) και των **προβλέψεων του νευρωνικού δικτύου** (πορτοκαλί διακεκομμένη καμπύλη) για τις τιμές της συνάρτησης y(t) σε διάφορα σημεία του t. Οι προβλέψεις του μοντέλου ακολουθούν πολύ κοντά την αναλυτική καμπύλη σε όλο το εύρος, με μικρές αποκλίσεις μόνο σε ελάχιστες τιμές του t. Η γενική συμπεριφορά των δύο καμπυλών είναι **ταυτόσημη**, γεγονός που δείχνει ότι το μοντέλο "έμαθε" σωστά τη σχέση μεταξύ t και y(t).

Συμπερασματικά, το νευρωνικό δίκτυο **παρήγαγε προβλέψεις υψηλής ακρίβειας**, οι οποίες συμφωνούν σχεδόν τέλεια με την πραγματική λύση. Οι μικρές διαφορές σε ορισμένα σημεία δεν επηρεάζουν την ορθότητα της συνολικής προσέγγισης, γεγονός που επιβεβαιώνει την **αποτελεσματικότητα της εκπαίδευσης** και την ικανότητα του μοντέλου να γενικεύει καλά.

5.5 Σύγκριση Πολλαπλών Βελτιστοποιητών (custom RK4,Full-Batch GD & Adam) για την Προσέγγιση Λύσης ΣΔΕ με Νευρωνικά Δίκτυα

Σε αυτήν την ενότητα, παρουσιάζεται η υλοποίηση ενός νευρωνικού δικτύου που χρησιμοποιεί τους 3 βελτιστοποιητές που αναλύθηκαν προηγουμένως, έτσι ώστε να μπορούμε να τους συγκρίνουμε, για την προσέγγιση της λύσης του ΠΑΤ της παρακάτω ΣΔΕ:

$$\frac{dy}{dt} = y^2 e^t - y, y(0) = 0.5$$

με αντίστοιχη αναλυτική λύση:

$$y(t) = \frac{1}{e^t(2-t)}$$

Ακολουθεί μια ενιαία επεξήγηση του κώδικα:

Ο κώδικας ακολουθεί μια οργανωμένη διαδικασία για την επίλυση μιας διαφορικής εξίσωσης πρώτης τάξης μέσω της εκπαίδευσης ενός νευρωνικού δικτύου, χρησιμοποιώντας διαφορετικούς βελτιστοποιητές. Αρχικά, εισάγονται οι απαραίτητες βιβλιοθήκες: η PyTorch για τη δημιουργία και εκπαίδευση του νευρωνικού δικτύου, η NumPy για τη δημιουργία των δεδομένων εκπαίδευσης και η Matplotlib για την οπτικοποίηση των αποτελεσμάτων.

Το νευρωνικό δίκτυο ορίζεται μέσω της κλάσης SimpleNN, που αποτελεί ένα πλήρως συνδεδεμένο πολυεπίπεδο δίκτυο (MLP). Το δίκτυο δέχεται ως είσοδο μία τιμή, διαθέτει ένα κρυφό επίπεδο με 10 νευρώνες και λειτουργία ενεργοποίησης ReLU, και αποδίδει μία τιμή εξόδου που προσεγγίζει τη λύση της διαφορικής εξίσωσης.

Η συνάρτηση κόστους που χρησιμοποιείται είναι το μέσο τετραγωνικό σφάλμα (MSE), το οποίο αξιολογεί την ακρίβεια των προβλέψεων συγκρίνοντάς τες με την αναλυτική λύση της εξίσωσης. Αυτό επιτρέπει την εκτίμηση της απόδοσης του μοντέλου, με σκοπό τη μείωση της απόκλισης μέσω της διαδικασίας εκπαίδευσης.

Η εκπαίδευση του μοντέλου γίνεται με τρεις διαφορετικούς βελτιστοποιητές:

- **Custom RK4**: Πρόκειται για μια προσαρμοσμένη υλοποίηση της μεθόδου Runge-Kutta τέταρτης τάξης. Σε κάθε βήμα, υπολογίζονται τέσσερις ενδιάμεσες κλίσεις (k1, k2, k3, k4) και το βάρος ενημερώνεται με βάση έναν συνδυασμό αυτών των κλίσεων. Αυτή η μέθοδος προσφέρει μεγαλύτερη ακρίβεια στις ενημερώσεις των παραμέτρων.

- **Full-Batch Gradient Descent (Full-Batch GD)**: Ενημερώνει τα βάρη χρησιμοποιώντας κλίσεις που προκύπτουν από ολόκληρο το dataset, εφαρμόζοντας έναν σταθερό ρυθμό μάθησης.
- **Adam**: Ένας εξελιγμένος βελτιστοποιητής που συνδυάζει την προσαρμογή των ρυθμών μάθησης και το momentum για ταχύτερη και πιο σταθερή σύγκλιση.

Τα δεδομένα εκπαίδευσης βασίζονται στην αναλυτική λύση της εξίσωσης, με τον ορισμό του πεδίου στο διάστημα [0,1.9], αποφεύγοντας το σημείο t=2, όπου η συνάρτηση καθίσταται μη ορισμένη. Δημιουργούνται 100 ομοιόμορφα κατανεμημένα σημεία, τα οποία μετατρέπονται σε PyTorch tensors για χρήση κατά την εκπαίδευση.

Η διαδικασία εκπαίδευσης εκτελείται για 2000 εποχές, χρησιμοποιώντας ίδιο ρυθμό μάθησης (0.1) για όλους τους βελτιστοποιητές. Κάθε εποχή περιλαμβάνει τα εξής στάδια:

1. **Forward Pass**: Υπολογισμός των προβλέψεων του δικτύου για κάθε τιμή εισόδου.
2. **Υπολογισμός Κόστους**: Σύγκριση των προβλέψεων με τις πραγματικές τιμές της αναλυτικής λύσης.
3. **Backward Pass**: Υπολογισμός των παραγώγων της συνάρτησης κόστους ως προς τα βάρη του δικτύου.
4. **Ενημέρωση Βαρών**: Αναπροσαρμογή των βαρών σύμφωνα με τον εκάστοτε βελτιστοποιητή.

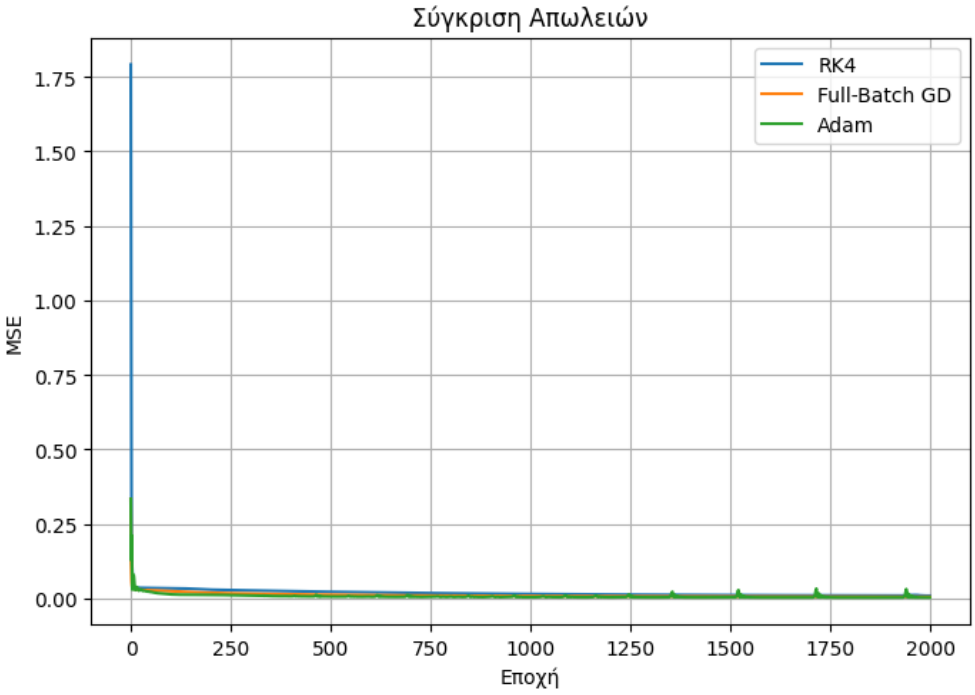
Για την αξιολόγηση της απόδοσης των βελτιστοποιητών, δημιουργούνται τέσσερα γραφήματα:

- **Καμπύλες Απώλειας**: Απεικονίζουν την εξέλιξη του μέσου τετραγωνικού σφάλματος κατά τη διάρκεια της εκπαίδευσης για κάθε βελτιστοποιητή. Το γράφημα επιτρέπει τη σύγκριση της ταχύτητας και της ομαλότητας σύγκλισης των διαφορετικών μεθόδων.
- **Προβλέψεις έναντι Αναλυτικής Λύσης**: Συγκρίνει τις προβλέψεις του νευρωνικού δικτύου με την αναλυτική λύση. Η αναλυτική λύση αποδίδεται με συνεχή μπλε γραμμή, ενώ οι προβλέψεις των βελτιστοποιητών απεικονίζονται με διακεκομμένες γραμμές διαφορετικών χρωμάτων.
- **Απόλυτο Σφάλμα (Γραμμική Κλίμακα)**: Δείχνει την απόλυτη διαφορά μεταξύ της πρόβλεψης και της πραγματικής τιμής για κάθε σημείο, αναδεικνύοντας τις περιοχές όπου το μοντέλο εμφανίζει τη μεγαλύτερη απόκλιση.
- **Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)**: Απεικονίζει το απόλυτο σφάλμα σε λογαριθμική κλίμακα, διευκολύνοντας την ανάλυση μικρών αποκλίσεων και προσφέροντας μια πιο λεπτομερή εικόνα της ακρίβειας του μοντέλου.

Η συνολική ανάλυση στοχεύει στην αξιολόγηση της αποτελεσματικότητας κάθε βελτιστοποιητή ως προς την ικανότητα του νευρωνικού δικτύου να προσεγγίσει την αναλυτική λύση με ακρίβεια και σταθερότητα.

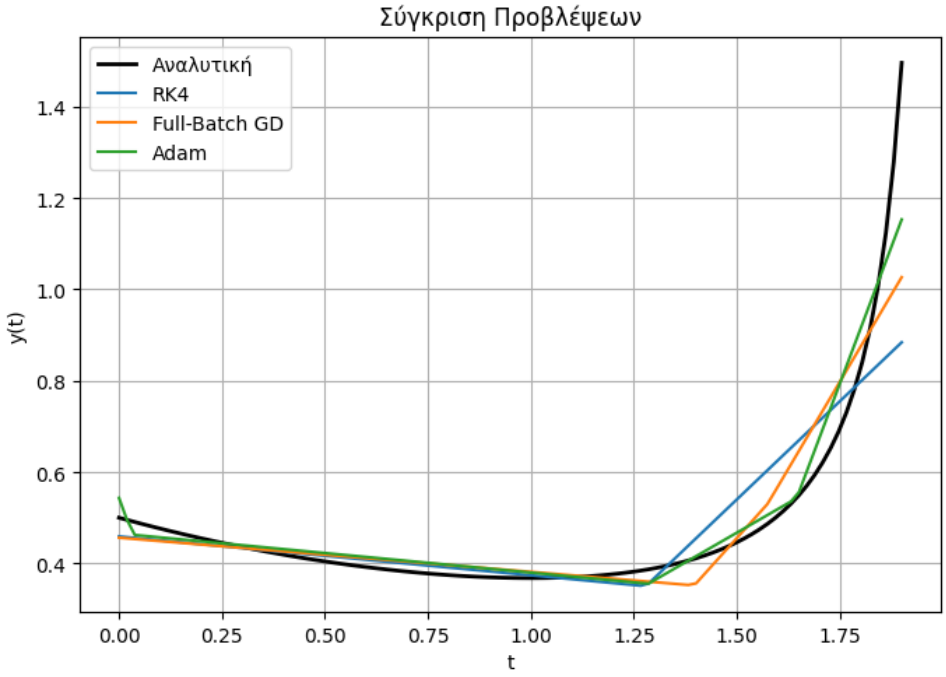
<u>Σημείωση</u>: Ο κώδικας βρίσκεται στο Παράρτημα Κώδικα!

Ακολουθεί επεξήγηση των γραφημάτων του κώδικα:



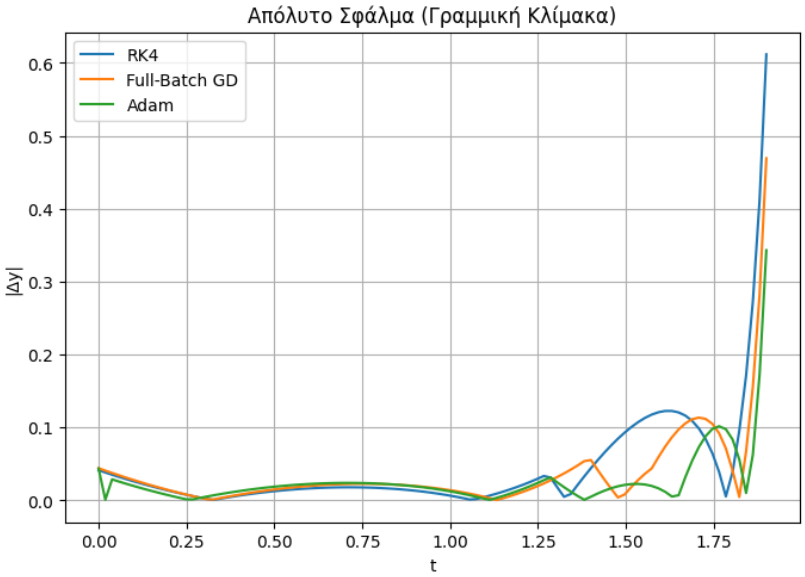
Σχήμα 5.10: Σύγκριση Απωλειών

Στο γράφημα "*Σύγκριση Απωλειών*" φαίνεται η **εξέλιξη του σφάλματος (MSE)** για τρεις μεθόδους (RK4, Full-Batch GD, Adam) καθώς αυξάνονται οι εποχές εκπαίδευσης (από 0 έως 2000). Όλες οι καμπύλες ξεκινούν από υψηλές τιμές και πέφτουν γρήγορα τις πρώτες ~100 εποχές. Ο **Adam** εμφανίζει **ταχύτερη αρχική πτώση** του σφάλματος σε σύγκριση με τις άλλες μεθόδους, ενώ ο RK4 φτάνει σε πολύ κοντινές τιμές με τον Full-Batch GD (σε συγκεκριμένα σημεία).



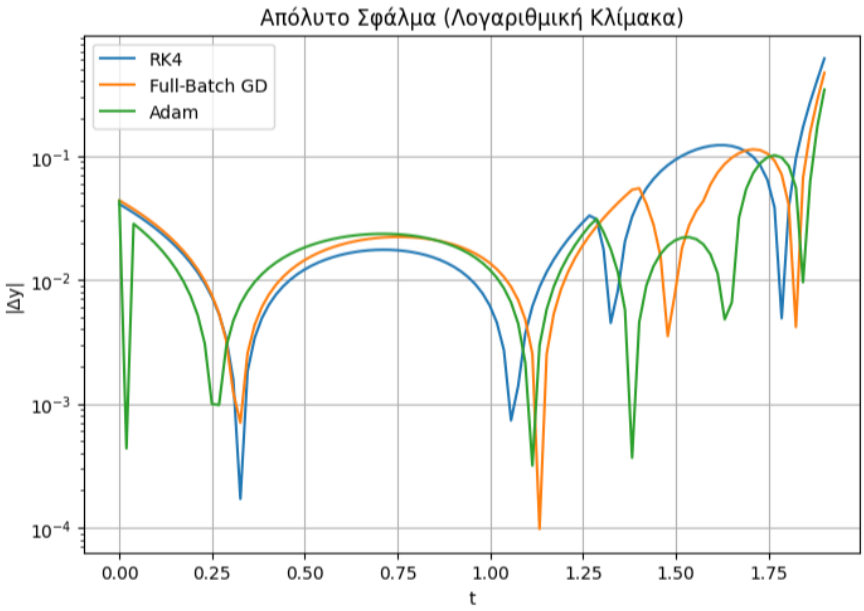
Σχήμα 5.11: Σύγκριση Προβλέψεων

Στο γράφημα "*Σύγκριση Προβλέψεων*", ο **Adam προσεγγίζει καλύτερα την αναλυτική λύση** σε σχέση με τους άλλους 2 βελτιστοποιητές (RK4 & Full-Batch GD). Συγκεκριμένα, οι δύο μέθοδοι (RK4 & Full-Batch GD) αποκλίνουν εμφανώς, ενώ ο Adam παραμένει αρκετά ακριβής.



Σχήμα 5.12: Απόλυτο Σφάλμα (Γραμμική Κλίμακα)

Το διάγραμμα "*Απόλυτο Σφάλμα (Γραμμική Κλίμακα)*" δείχνει ότι ο **Adam** έχει **καλύτερο απόλυτο σφάλμα** σε σύγκριση με τον Full-Batch GD, αλλά και τον RK4. Ο Adam εμφανίζει ελαφρές διακυμάνσεις, αλλά διατηρεί καλύτερη απόδοση και από τους 2 άλλους βελτιστοποιητές.



Σχήμα 5.13: Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)

Στη λογαριθμική κλίμακα, φαίνεται ότι ο **Adam έχει πιο απότομη πτώση** του σφάλματος στις αρχικές τιμές του t σε σύγκριση με τους άλλους 2 βελτιστοποιητές, ακολουθώντας μια εκθετική μείωση. Ωστόσο, σε μερικά σημεία το σφάλμα και των 3 βελτιστοποιητών υπερτερεί.

Γενικό Συμπέρασμα: Ο **Adam είναι καλύτερος από τον RK4 & τον Full-Batch GD** (κάτι το οποίο είναι αναμενόμενο) τόσο σε ταχύτητα σύγκλισης όσο και σε απόδοση, ιδίως σε σύντομα χρονικά διαστήματα. Ωστόσο, οι άλλοι 2 βελτιστοποιητές έχουν περίπου παρόμοια συμπεριφορά με βάση τα γραφήματα!

Ωστόσο, δεν γίνεται δίκαιη σύγκριση μεταξύ των 3 βελτιστοποιητών!

Γιατί δεν αποτελεί δίκαιη σύγκριση;

Το σημαντικότερο πρόβλημα είναι η χρήση Κοινού Learning Rate (LR)

Οι βελτιστοποιητές έχουν διαφορετικές απαιτήσεις ως προς το learning rate:

- **RK4**: Είναι ευαίσθητος στο h (το οποίο λειτουργεί ως learning rate), καθώς χρησιμοποιεί πολλαπλά βήματα για την ενημέρωση των παραμέτρων.
- **Full-Batch GD**: Απαιτεί συνήθως μικρότερο learning rate για να συγκλίνει σταθερά.
- **Adam**: Είναι πιο ανεκτικός σε μεγαλύτερες τιμές learning rate, καθώς χρησιμοποιεί εκθετικά μέσα για την προσαρμογή των κλίσεων.

Πρόβλημα: Χρησιμοποιώντας το ίδιο LR = 0.1 για όλους τους βελτιστοποιητές, δεν λαμβάνεται υπόψη η διαφορετική συμπεριφορά κάθε βελτιστοποιητή.

5.6 Σύγκριση Πολλαπλών Βελτιστοποιητών (custom RK4,Full-Batch GD & Adam) για την Προσέγγιση Λύσης ΣΔΕ με Νευρωνικά Δίκτυα (με τα πιθανά καλύτερα h/lr για κάθε βελτιστοποιητή)

Σε αυτήν την ενότητα, παρουσιάζεται η υλοποίηση ενός νευρωνικού δικτύου που χρησιμοποιεί τους 3 βελτιστοποιητές που αναλύθηκαν προηγουμένως (*αλλά με τα πιθανά καλύτερα h/lr για κάθε βελτιστοποιητή*), έτσι ώστε να μπορούμε να τους συγκρίνουμε, για την προσέγγιση της λύσης του ΠΑΤ της παρακάτω ΣΔΕ:

$$\frac{dy}{dt} = y^2 e^t - y, y(0) = 0.5$$

με αντίστοιχη αναλυτική λύση:

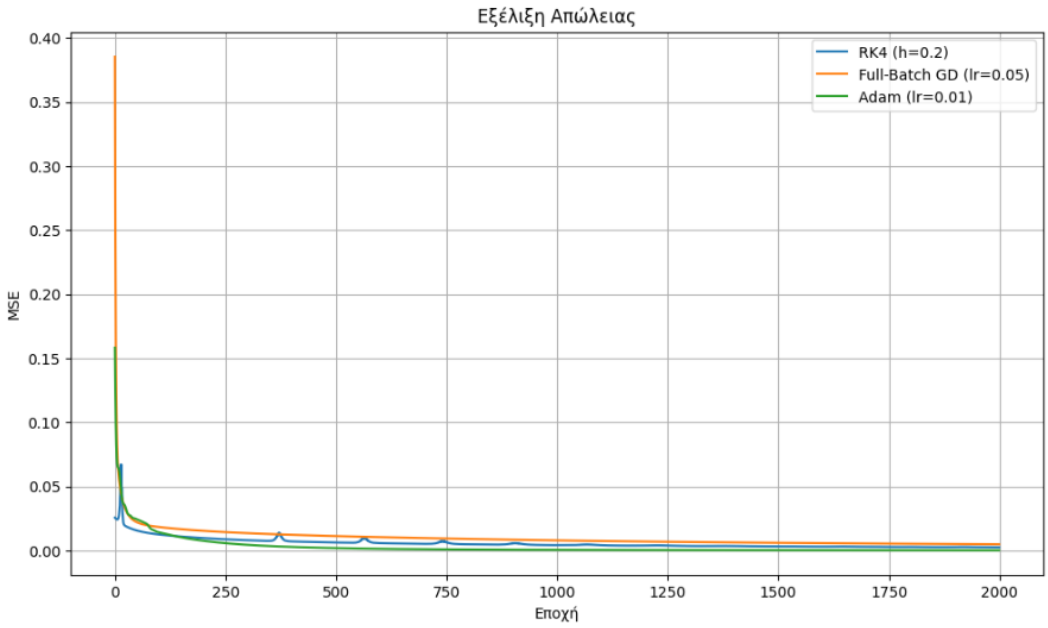
$$y(t) = \frac{1}{e^t(2-t)}$$

Ακολουθεί μια συνοπτική επεξήγηση της αλλαγής του κώδικα της ενότητας 5.5 για να γίνουν δοκιμές για τον κώδικα της ενότητας 5.6:

Ο κώδικας βρίσκεται στην ίδια λογική με αυτόν την ενότητας 5.5, απλώς εφαρμόζουμε Grid Search για την επιλογή των καλύτερων δυνατών παραμέτρων. Οι τιμές του **h** (δηλαδή για τον βελτιστοποιητή **custom RK4**) είναι: [0.01, 0.05, 0.2, 0.3, 0.4, 0.5] ενώ για το **lr** (δηλαδή για τους βελτιστοποιητές **Adam** & **Full-Batch GD**) είναι: [0.0001, 0.001, 0.005, 0.01, 0.05, 0.5].

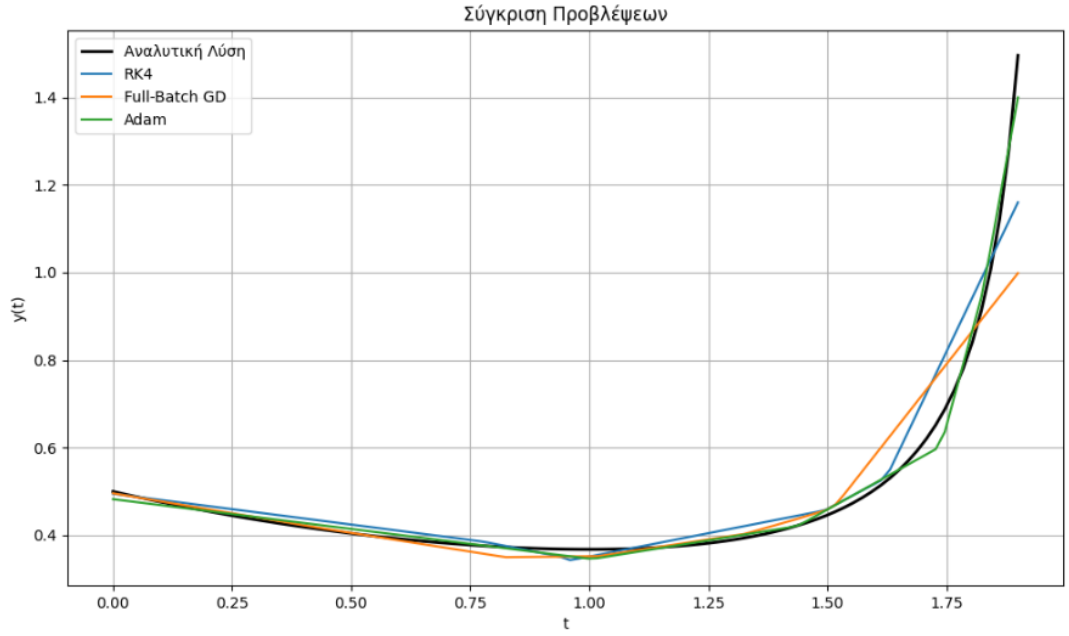
<u>Σημείωση</u>: Ο κώδικας βρίσκεται στο Παράρτημα Κώδικα!

Ακολουθεί επεξήγηση των γραφημάτων του κώδικα:



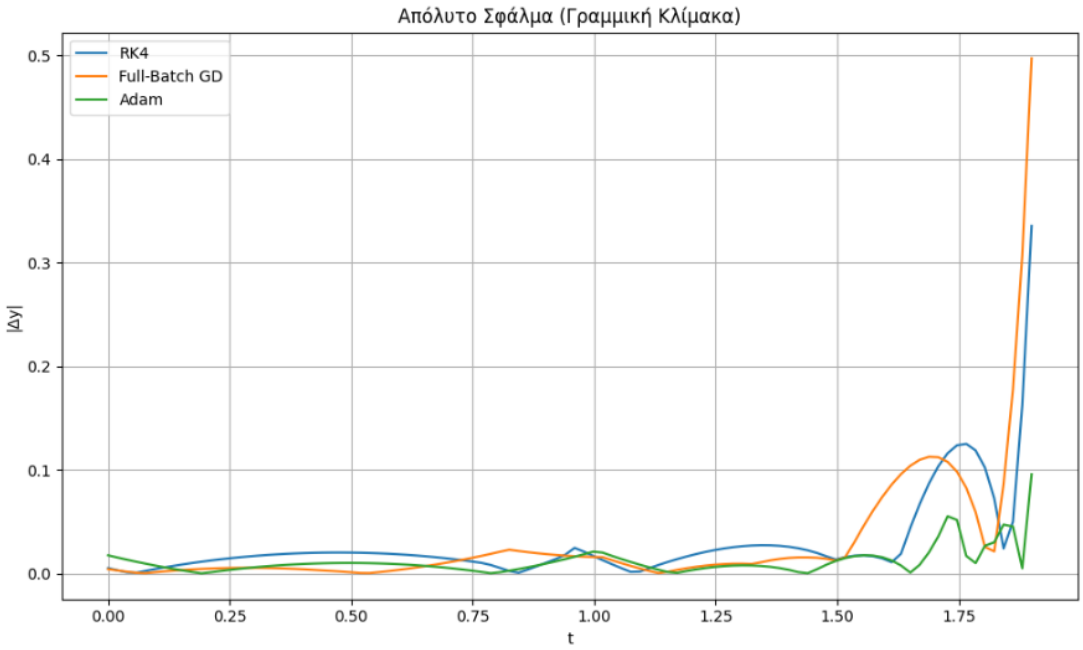
Σχήμα 5.14: Εξέλιξη Απώλειας

Στο γράφημα "*Εξέλιξη Απώλειας*" παρατηρείται η συμπεριφορά των τριών μεθόδων βελτιστοποίησης (**RK4 με h=0.2**, **Full-Batch GD με lr=0.05** & **Adam με lr=0.01**) σε σχέση με τη μέση τετραγωνική απώλεια (MSE) κατά τη διάρκεια 2000 εποχών εκπαίδευσης. Ο **RK4**, χάρη στο μεγάλο βήμα (h=0.2), επιτυγχάνει σταθερή και γρήγορη μείωση της απώλειας, φτάνοντας σε επίπεδα κοντά στο μηδέν, ωστόσο η αποτελεσματικότητά του εξαρτάται από ακριβή ρύθμιση της παραμέτρου h. Ο **Full-Batch GD**, παρά τον ρυθμό μάθησης 0.05, εμφανίζει αργή βελτίωση με πιθανές διακυμάνσεις, πράγμα που υποδηλώνει δυσκολίες στην προσέγγιση του βέλτιστου σημείου. Ο **Adam**, με χαμηλότερο ρυθμό μάθησης (lr=0.01), με γρήγορη πτώση της απώλειας από τις πρώτες εποχές και σταθερή πορεία χωρίς έντονες αναταράξεις, επιδεικνύοντας ισορροπία μεταξύ ταχύτητας και σταθερότητας. Σε πρακτικό πλαίσιο, ο **Adam** διατηρεί το πλεονέκτημα λόγω ευελιξίας και αξιοπιστίας, ενώ ο **RK4** μπορεί να αξιοποιηθεί σε ειδικές περιπτώσεις με προσεκτική παραμετροποίηση. Ο **Full-Batch GD**, από την άλλη, φαίνεται να απαιτεί περαιτέρω βελτιστοποίηση για να ανταποκριθεί αποτελεσματικά σε πραγματικές απαιτήσεις.



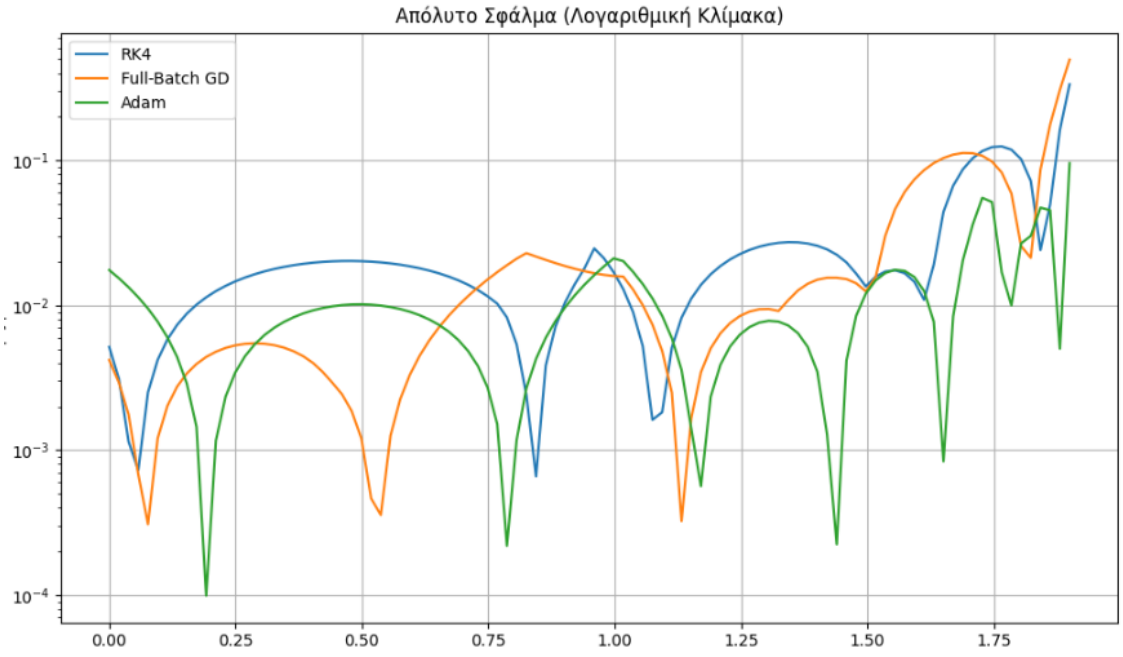
Σχήμα 5.15: Σύγκριση Προβλέψεων

Στο γράφημα "*Σύγκριση Προβλέψεων*" εξετάζεται η προσέγγιση τριών μεθόδων (**RK4**, **Full-Batch GD**, **Adam**) στην **Αναλυτική Λύση**, η οποία αντιπροσωπεύει το ιδανικό αποτέλεσμα, σε ένα εύρος τιμών από 0.00 έως 1.9. Η **Αναλυτική Λύση** παρουσιάζει μια σταθερή και ακριβής τάση, χρησιμεύοντας ως βάση σύγκρισης. Ο **RK4** εμφανίζει προβλέψεις που ακολουθούν στενά την αναλυτική λύση σε όλο το εύρος, υποδηλώνοντας υψηλή ακρίβεια και αξιοπιστία. Ο **Full-Batch GD** παρουσιάζει μεγαλύτερες αποκλίσεις, ιδίως σε υψηλότερες τιμές (π.χ. κοντά στο 1.9), πιθανώς λόγω δυσκολιών στη βελτιστοποίηση ή περιορισμών στη ρύθμιση παραμέτρων. Ο **Adam**, από την άλλη, διατηρεί μια ισορροπημένη προσέγγιση, με ελάχιστες αποκλίσεις από την αναλυτική λύση, γεγονός που επιβεβαιώνει την αποτελεσματικότητά του σε πραγματικά προβλήματα. Συνολικά, ο **Adam** για ακριβείς προσομοιώσεις, ο **RK4** για ευέλικτες και σταθερές προβλέψεις, ενώ ο **Full-Batch GD** απαιτεί περαιτέρω βελτιστοποίηση για να ανταγωνιστεί. Ουσιαστικά, η επιλογή εξαρτάται από τις απαιτήσεις ακρίβειας, τα διαθέσιμα δεδομένα και τους υπολογιστικούς πόρους.



Σχήμα 5.16: Απόλυτο Σφάλμα (Γραμμική Κλίμακα)

Στο γράφημα "*Απόλυτο Σφάλμα (Γραμμική Κλίμακα)*" συγκρίνεται η ακρίβεια τριών μεθόδων (**RK4**, **Full-Batch GD**, **Adam**) σε ένα εύρος τιμών από 0.00 έως 1.9. Ο **Adam** με το μικρότερο απόλυτο σφάλμα σε όλο το εύρος, γεγονός που το καθιστά ιδανικό για εφαρμογές υψηλής ακρίβειας. Ο **Full-Batch GD** παρουσιάζει μεγαλύτερο σφάλμα, ιδίως σε υψηλότερες τιμές (π.χ. κοντά στο 1.9), πιθανώς λόγω δυσκολιών στη βελτιστοποίηση ή μη ιδανικής ρύθμισης. Ο **RK4**, από την άλλη, διατηρεί σφάλμα χαμηλά, με ελαφρώς μεγαλύτερες αποκλίσεις σε ορισμένα σημεία, αλλά διατηρεί ισορροπία μεταξύ ταχύτητας και ακρίβειας. Συνολικά, η επιλογή εξαρτάται από τις απαιτήσεις: ο **Adam** για μέγιστη ακρίβεια, ο **RK4** για ευελιξία και ο **Full-Batch GD** μόνο εάν υπάρχει δυνατότητα περαιτέρω βελτιστοποίησης.



Σχήμα 5.16: Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)

Στο γράφημα "*Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)*" συγκρίνεται η ακρίβεια των μεθόδων **RK4**, **Full-Batch GD** και **Adam** σε ένα εύρος τιμών από 0.00 έως 1.9. Η χρήση λογαριθμικής κλίμακας επισημαίνει τις μικρές διαφορές στο σφάλμα, ειδικά σε χαμηλά επίπεδα. Ο **Adam** με το μικρότερο απόλυτο σφάλμα σε όλο το εύρος, επιβεβαιώνοντας την υψηλή του ακρίβεια, ιδανική για εφαρμογές όπου η ακριβής προσέγγιση είναι κρίσιμη. Ο **Full-Batch GD** εμφανίζει σημαντικά μεγαλύτερο σφάλμα, ιδίως σε τιμές πάνω από 1.25, πιθανώς λόγω δυσκολιών στη βελτιστοποίηση ή μη ιδανικής ρύθμισης. Ο **RK4**, από την άλλη, διατηρεί σφάλμα κοντά στις χαμηλές τιμές, με ελαφρώς αυξημένες αποκλίσεις σε υψηλότερα επίπεδα, διατηρώντας όμως ισορροπία μεταξύ ταχύτητας και σταθερότητας.

6 Συμπεράσματα & Μελλοντικές Επεκτάσεις

6.1 Συμπεράσματα

Στην παρούσα πτυχιακή εργασία, διερευνήθηκε πρακτικά η χρήση της μεθόδου Runge-Kutta 4ης τάξης (RK4) ως βελτιστοποιητής για την εκπαίδευση νευρωνικών δικτύων σε σύγκριση με καθιερωμένους βελτιστοποιητές, όπως ο Full-Batch Gradient Descent (Full-Batch GD) και ο Adam. Αρχικά, ο custom RK4 βελτιστοποιητής έδειξε υψηλή ακρίβεια στην ενημέρωση των παραμέτρων του δικτύου, προσφέροντας σταθερή και γρήγορη μείωση της απώλειας κατά τη διάρκεια της εκπαίδευσης. Ωστόσο, η αποτελεσματικότητα του εξαρτάται σημαντικά από την ακριβή ρύθμιση της παραμέτρου h (μέγεθος βήματος).

Ωστόσο, ο βελτιστοποιητής Adam επικράτησε σε σύγκριση με τον RK4 και τον Full-Batch GD, καθώς επέδειξε ταχύτερη σύγκλιση και μικρότερο απόλυτο σφάλμα σε περισσότερα σημεία. Ο Adam διατήρησε μια ισορροπημένη προσέγγιση μεταξύ ταχύτητας και σταθερότητας, κάτι που τον καθιστά ιδιαίτερα κατάλληλο για πρακτικές εφαρμογές. Από την άλλη πλευρά, ο Full-Batch GD παρουσίασε αργή βελτίωση και μεγαλύτερες διακυμάνσεις, πράγμα που υποδηλώνει δυσκολίες στην προσέγγιση του βέλτιστου σημείου, εκτός εάν χρησιμοποιηθεί κατάλληλος ρυθμός μάθησης.

Τελικά, παρά την υψηλή ακρίβεια της RK4, ο Adam αποδείχθηκε καλύτερος σε περισσότερες περιπτώσεις, ενώ ο Full-Batch GD μπορεί να είναι ανταγωνιστικός μόνο όταν χρησιμοποιείται ένα κατάλληλο lr. Συνεπώς, η επιλογή του βελτιστοποιητή εξαρτάται από τις ανάγκες του προβλήματος: ο Adam είναι ιδανικός για εφαρμογές υψηλής ακρίβειας, ο RK4 για ευελιξία και σταθερότητα, ενώ ο Full-Batch GD απαιτεί περαιτέρω βελτιστοποίηση για να είναι ανταγωνιστικός.

Συμπερασματικά, η εφαρμογή της μεθόδου Runge-Kutta στα νευρωνικά δίκτυα αποτελεί μια υποσχόμενη προσέγγιση που μπορεί να βελτιώσει σημαντικά την απόδοση και την ακρίβεια των μοντέλων. Ωστόσο, η περαιτέρω έρευνα είναι απαραίτητη για να ξεπεραστούν οι υπάρχοντες περιορισμοί και να διερευνηθούν νέες εφαρμογές. Η συνδυασμένη χρήση της RK4 με άλλες τεχνικές και η επέκταση της σε νέους τομείς μπορεί να ανοίξει νέους δρόμους για την ανάπτυξη πιο αποτελεσματικών και ακριβών μοντέλων τεχνητής νοημοσύνης.

6.2 Μελλοντική Έρευνα

Βελτιστοποίηση της RK4:

- Μελέτη και ανάπτυξη αυτόματων μεθόδων για την επιλογή της βέλτιστης τιμής του *h* (μέγεθος βήματος) για διαφορετικά προβλήματα.
- Εξέταση της ενσωμάτωσης της RK4 σε πιο πολύπλοκες αρχιτεκτονικές νευρωνικών δικτύων, όπως τα Recurrent Neural Networks (RNNs) και Convolutional Neural Networks (CNNs).

Συνδυασμός με άλλες τεχνικές:

- Διερεύνηση της παράλληλης χρήσης της RK4 με άλλους βελτιστοποιητές (π.χ., Adam) για να εκμεταλλευτούμε τα πλεονεκτήματα και των δύο προσεγγίσεων.
- Ανάπτυξη υβριδικών μεθόδων που συνδυάζουν την RK4 με τεχνικές όπως το dropout ή η κανονικοποίηση για βελτίωση της γενίκευσης.

Εφαρμογές σε πραγματικά προβλήματα:

- Εφαρμογή της RK4 σε πραγματικά προβλήματα, όπως η ανάλυση ιατρικών εικόνων, η πρόβλεψη χρονικών σειρών, η ρομποτική και η υπολογιστική φυσική.
- Μελέτη της απόδοσης της RK4 σε προβλήματα με μεγάλα δεδομένα και σύνθετες δυναμικές.

Επέκταση σε άλλους τομείς:

- Εξέταση της χρήσης της RK4 σε άλλους τομείς της τεχνητής νοημοσύνης, όπως η ενισχυτική μάθηση (Reinforcement Learning) και η εξαγωγή γνώσης από δεδομένα (Knowledge Extraction).
- Ανάπτυξη νέων μεθόδων που βασίζονται στην RK4 για τη μείωση της πολυπλοκότητας μαθηματικών μοντέλων.

Υπολογιστική Αποδοτικότητα:

- Έρευνα για την επιτάχυνση των υπολογισμών της RK4 μέσω παράλληλης επεξεργασίας ή χρήσης γραφικών επεξεργαστών (GPUs).
- Εξέταση της δυνατότητας χρήσης προσεγγιστικών μεθόδων για τη μείωση του υπολογιστικού κόστους της RK4 χωρίς να θυσιάζεται η ακρίβεια.

Θεωρητική Ανάλυση:

- Περαιτέρω μελέτη της σταθερότητας και της σύγκλισης της RK4 σε διάφορα προβλήματα, ιδιαίτερα σε άκαμπτα συστήματα (stiff systems).
- Ανάπτυξη νέων κριτηρίων ευστάθειας για την RK4 και άλλες μεθόδους Runge-Kutta

Παράρτημα Κώδικα

Σύγκριση μεταξύ των μεθόδων Runge Kutta (RK1/RK2/RK3/RK4)

```
#
================================================================
========

# Εισαγωγή Βιβλιοθηκών

#
================================================================
========

import numpy as np  # Βιβλιοθήκη για επιστημονικούς υπολογισμούς

import matplotlib.pyplot as plt  # Βιβλιοθήκη για δημιουργία γραφημάτων


#
================================================================
========

# Ορισμός της Διαφορικής Εξίσωσης Bernoulli (dy/dt = y^2 e^t - y)

#
================================================================
========

def f(t, y):

    """Υπολογίζει την παράγωγο dy/dt για τη διαφορική εξίσωση.


    Παράμετροι:

        t (float): Τρέχων χρόνος

        y (float): Τρέχουσα τιμή της συνάρτησης y(t)


    Επιστρέφει:

        float: Τη τιμή της παραγώγου dy/dt

    """

    return y**2 * np.exp(t) - y  # Επιστροφή της μαθηματικής έκφρασης


#
================================================================
========

# Αναλυτική Λύση της Διαφορικής Εξίσωσης
```

```
#
================================================================================

def exact(t):
    """Επιστρέφει την ακριβή αναλυτική λύση y(t) που προκύπτει από την ολοκλήρωση της εξίσωσης.

    **Σημαντικό**: Η λύση ισχύει μόνο για t<2 (ο παρονομαστής γίνεται 0 στο t=2).

    Παράμετροι:
        t (float ή np.ndarray): Χρόνος για τους οποίους ζητείται η λύση

    Επιστρέφει:
        float ή np.ndarray: Τη τιμή της αναλυτικής λύσης
    """
    return 1 / (np.exp(t) * (2 - t))  # Τύπος λύσης που βρέθηκε αναλυτικά

#
================================================================================
# Μέθοδος Runge-Kutta 1ης Τάξης (RK1 - Euler)
#
================================================================================
def rk1(f, y0, t0, tf, h):
    """Επίλυση ΣΔΕ με τη μέθοδο Euler (RK1).

    Παράμετροι:
        f (function): Συνάρτηση της παραγώγου dy/dt = f(t, y)
        y0 (float): Αρχική συνθήκη y(t0)
        t0 (float): Αρχικός χρόνος
        tf (float): Τελικός χρόνος
        h (float): Βήμα διακριτοποίησης

    Επιστρέφει:
```

```
        tuple: (πίνακας χρόνων t, πίνακας λύσεων y)
    """
    t = np.arange(t0, tf + h, h)  # Δημιουργία πίνακα χρόνων [t0, t0+h,
..., tf]
    y = np.zeros(len(t))  # Πίνακας για τις προσεγγίσεις της λύσης
    y[0] = y0  # Εφαρμογή αρχικής συνθήκης

    # Επανάληψη για κάθε βήμα χρόνου
    for i in range(len(t)-1):
        y[i+1] = y[i] + h * f(t[i], y[i])  # Τύπος Euler: y_{n+1} =
y_n + h*f(t_n, y_n)

    return t, y  # Επιστροφή χρόνων και προσεγγίσεων

#
================================================================
========
# Μέθοδος Runge-Kutta 2ης Τάξης (RK2 - Midpoint)
#
================================================================
========
def rk2(f, y0, t0, tf, h):
    """Επίλυση ΣΔΕ με τη μέθοδο Midpoint (RK2).

    Χρησιμοποιεί ενδιάμεσο σημείο για ακριβέστερη προσέγγιση.

    Παράμετροι:
        f (function): Συνάρτηση της παραγώγου dy/dt = f(t, y)
        y0 (float): Αρχική συνθήκη y(t0)
        t0 (float): Αρχικός χρόνος
        tf (float): Τελικός χρόνος
        h (float): Βήμα διακριτοποίησης

    Επιστρέφει:
        tuple: (πίνακας χρόνων t, πίνακας λύσεων y)
    """
```

```
    t = np.arange(t0, tf + h, h)  # Δημιουργία πίνακα χρόνων [t0, t0+h,
..., tf]
    y = np.zeros(len(t))  # Πίνακας για τις προσεγγίσεις της λύσης
    y[0] = y0  # Εφαρμογή αρχικής συνθήκης

    # Επανάληψη για κάθε βήμα χρόνου
    for i in range(len(t)-1):
        k1 = h * f(t[i], y[i])  # Πρώτη κλίση (όπως στον Euler)
        k2 = h * f(t[i] + h/2, y[i] + k1/2)  # Δεύτερη κλίση στο μέσο
του βήματος
        y[i+1] = y[i] + k2  # Ενημέρωση με βάση τη δεύτερη κλίση

    return t, y  # Επιστροφή χρόνων και προσεγγίσεων

#
================================================================
========
# Μέθοδος Runge-Kutta 3ης Τάξης (RK3)
#
================================================================
========
def rk3(f, y0, t0, tf, h):
    """Επίλυση ΣΔΕ με τη μέθοδο RK3.

    Χρησιμοποιεί τρία βήματα για να πετύχει ακρίβεια τρίτης τάξης.

        Παράμετροι:
        f (function): Συνάρτηση της παραγώγου dy/dt = f(t, y)
        y0 (float): Αρχική συνθήκη y(t0)
        t0 (float): Αρχικός χρόνος
        tf (float): Τελικός χρόνος
        h (float): Βήμα διακριτοποίησης

    Επιστρέφει:
        tuple: (πίνακας χρόνων t, πίνακας λύσεων y)
    """
```

```
    t = np.arange(t0, tf + h, h)  # Δημιουργία πίνακα χρόνων [t0, t0+h,
..., tf]
    y = np.zeros(len(t))  # Πίνακας για τις προσεγγίσεις της λύσης
    y[0] = y0  # Εφαρμογή αρχικής συνθήκης

    # Επανάληψη για κάθε βήμα χρόνου
    for i in range(len(t)-1):
        k1 = h * f(t[i], y[i])  # Πρώτη κλίση (όπως στον Euler)
        k2 = h * f(t[i] + h/2, y[i] + k1/2)  # Πρώτο ενδιάμεσο βήμα
        k3 = h * f(t[i] + h, y[i] - k1 + 2*k2)  # Δεύτερο ενδιάμεσο
βήμα
        y[i+1] = y[i] + (k1 + 4*k2 + k3)/6  # Συνδυασμός βημάτων με
βάρη

    return t, y

#
================================================================
========
# Μέθοδος Runge-Kutta 4ης Τάξης (RK4 Method)
#
================================================================
========
def rk4(f, y0, t0, tf, h):
    """Επίλυση ΣΔΕ με τη μέθοδο Runge Kutta (RK4) .

    Η πιο διαδεδομένη μέθοδος για ισορροπία ακρίβειας-πολυπλοκότητας.

    Παράμετροι:
        f (function): Συνάρτηση της παραγώγου dy/dt = f(t, y)
        y0 (float): Αρχική συνθήκη y(t0)
        t0 (float): Αρχικός χρόνος
        tf (float): Τελικός χρόνος
        h (float): Βήμα διακριτοποίησης

    Επιστρέφει:
        tuple: (πίνακας χρόνων t, πίνακας λύσεων y)
```

```
    """
    t = np.arange(t0, tf + h, h)  # Δημιουργία πίνακα χρόνων [t0, t0+h, ..., tf]
    y = np.zeros(len(t))  # Πίνακας για τις προσεγγίσεις της λύσης
    y[0] = y0  # Εφαρμογή αρχικής συνθήκης

    # Επανάληψη για κάθε βήμα χρόνου
    for i in range(len(t)-1):
        k1 = h * f(t[i], y[i])  # Πρώτη κλίση (όπως Euler)
        k2 = h * f(t[i] + h/2, y[i] + k1/2)  # Δεύτερη κλίση στο μέσο με k1
        k3 = h * f(t[i] + h/2, y[i] + k2/2)  # Τρίτη κλίση στο μέσο με k2
        k4 = h * f(t[i] + h, y[i] + k3)  # Τέταρτη κλίση στο τέλος του βήματος
        y[i+1] = y[i] + (k1 + 2*k2 + 2*k3 + k4)/6  # Συνδυασμός με βάρη 1-2-2-1

    return t, y

# ================================================================================

# Ορισμός Παραμέτρων Προβλήματος
# ================================================================================

y0 = 0.5   # Αρχική συνθήκη y(0) = 0.5 (από την ακριβή λύση στο t=0)
t0 = 0     # Αρχικός χρόνος
tf = 1.9    # Τελικός χρόνος (αποφυγή ιδιομορφίας στο t=2 όπου ο παρονομαστής μηδενίζεται)
h = 0.1    # Βήμα διακριτοποίησης - Μικρό βήμα δίνει καλύτερη ακρίβεια αλλά περισσότερους υπολογισμούς

# Επίλυση με όλες τις μεθόδους
t_rk1, y_rk1 = rk1(f, y0, t0, tf, h)  # RK1
t_rk2, y_rk2 = rk2(f, y0, t0, tf, h)  # RK2
t_rk3, y_rk3 = rk3(f, y0, t0, tf, h)  # RK3
t_rk4, y_rk4 = rk4(f, y0, t0, tf, h)  # RK4
```

```
# Υπολογισμός αναλυτικής λύσης στους ίδιους χρόνους με το RK1
t_exact = t_rk1  # Χρήση των ίδιων χρονικών στιγμών για εύκολη σύγκριση
y_exact = exact(t_exact)  # Υπολογισμός ακριβών τιμών

#
================================================================
========

# Οπτικοποίηση Αποτελεσμάτων - Σύγκριση Λύσεων
#
================================================================
========
plt.figure(figsize=(12, 6))   # Δημιουργία figure με συγκεκριμένο
μέγεθος

# Plot αναλυτικής λύσης
plt.plot(t_exact, y_exact, 'k-', linewidth=2, label='Αναλυτική')   #
Μαύρη συνεχής γραμμή

# Plot προσεγγιστικών λύσεων με διαφορετικά markers
plt.plot(t_rk1, y_rk1, 'bo--', markersize=5, label='RK1')  # RK1
plt.plot(t_rk2, y_rk2, 'g^--', markersize=5, label='RK2')  # RK2
plt.plot(t_rk3, y_rk3, 'rs--', markersize=5, label='RK3')  # RK3
plt.plot(t_rk4, y_rk4, 'cd--', markersize=5, label='RK4')  # RK4

# Διαμόρφωση γραφήματος
plt.title('Σύγκριση Ακρίβειας Μεθόδων Runge-Kutta (RK1-RK4) με την
Αναλυτική Λύση')  # Τίτλος γραφήματος
plt.xlabel('t')  # Ετικέτα άξονα x
plt.ylabel('y(t)')  # Ετικέτα άξονα y
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος

#
================================================================
========

# Οπτικοποίηση Σφαλμάτων (Γραμμική Κλίμακα)
```

```
#
================================================================
========

plt.figure(figsize=(12, 6))    # Δημιουργία figure με συγκεκριμένο
μέγεθος


# Υπολογισμός απόλυτων σφαλμάτων για κάθε μέθοδο

plt.plot(t_exact, np.abs(y_exact - y_rk1), 'bo--', markersize=5,
label='RK1 Σφάλμα')  # RK1

plt.plot(t_exact, np.abs(y_exact - y_rk2), 'g^--', markersize=5,
label='RK2 Σφάλμα')  # RK2

plt.plot(t_exact, np.abs(y_exact - y_rk3), 'rs--', markersize=5,
label='RK3 Σφάλμα')  # RK3

plt.plot(t_exact, np.abs(y_exact - y_rk4), 'cd--', markersize=5,
label='RK4 Σφάλμα')  # RK4


# Διαμόρφωση γραφήματος

plt.title('Απόλυτο Σφάλμα (Γραμμική Κλίμακα)')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('|Αναλυτική - Αριθμητική|')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος

plt.show()  # Εμφάνιση γραφήματος


#
================================================================
========

# Οπτικοποίηση Σφαλμάτων (Λογαριθμική Κλίμακα)

#
================================================================
========

plt.figure(figsize=(12, 6))    # Δημιουργία figure με συγκεκριμένο
μέγεθος


# Χρήση semilogy για λογαριθμικό άξονα y

plt.semilogy(t_exact, np.abs(y_exact - y_rk1), 'bo--', markersize=5,
label='RK1 Σφάλμα')  # RK1

plt.semilogy(t_exact, np.abs(y_exact - y_rk2), 'g^--', markersize=5,
label='RK2 Σφάλμα')  # RK2
```

```
plt.semilogy(t_exact, np.abs(y_exact - y_rk3), 'rs--', markersize=5, label='RK3 Σφάλμα')  # RK3
plt.semilogy(t_exact, np.abs(y_exact - y_rk4), 'cd--', markersize=5, label='RK4 Σφάλμα')  # RK4


# Διαμόρφωση γραφήματος
plt.title('Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)')  # Τίτλος γραφήματος
plt.xlabel('t')  # Ετικέτα άξονα x
plt.ylabel('|Αναλυτική - Αριθμητική|')  # Ετικέτα άξονα y
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος
```

Υλοποίηση ενός Νευρωνικού Δικτύου για την προσέγγιση λύσης ΣΔΕ (με τη χρήση του αριθμητικού αλγορίθμου Runge-Kutta 4ου βαθμού (RK4) ως προσαρμοσμένου βελτιστοποιητή)

```
# ------------- Εισαγωγή Βιβλιοθηκών -------------
import torch  # Βασική βιβλιοθήκη για δουλειά με tensors και νευρωνικά δίκτυα
import torch.nn as nn  # Επίπεδα νευρωνικών δικτύων (π.χ. Linear, ReLU)
import numpy as np  # Βιβλιοθήκη για επιστημονικούς υπολογισμούς
import matplotlib.pyplot as plt  # Βιβλιοθήκη για δημιουργία γραφημάτων
import contextlib  # Βιβλιοθήκη για τη διαχείριση context managers

# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Ορισμός Νευρωνικού Δικτύου -------------
class SimpleNN(nn.Module):
    def __init__(self):
        """
        Αρχικοποιεί ένα δίκτυο με:
        - fc1: Πρώτο γραμμικό επίπεδο (1 είσοδος -> 10 νευρώνες)
```

```
        - fc2: Δεύτερο γραμμικό επίπεδο (10 νευρώνες -> 1 έξοδος)
        """
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(
            in_features=1,  # Διαστάσεις εισόδου
            out_features=10  # Αριθμός νευρώνων στο κρυφό επίπεδο
        )
        self.fc2 = nn.Linear(
            in_features=10,  # Είσοδος από τους 10 νευρώνες του fc1
            out_features=1  # Έξοδος (πρόβλεψη)
        )

    def forward(self, x):
        """
        Forward pass του δικτύου:
        - x: Tensor εισόδου με shape (N, 1), όπου N = αριθμός δειγμάτων
        - Επιστρέφει: Tensor εξόδου με shape (N, 1)
        """
         x = torch.relu(self.fc1(x))  # ReLU ενεργοποίηση στο κρυφό
επίπεδο
        x = self.fc2(x)  # Γραμμικός μετασχηματισμός στην έξοδο
        return x


# ------------- Συνάρτηση Κόστους (MSE) -------------
def loss_fn(y_pred, y_true):
    """
    Υπολογίζει το Μέσο Τετραγωνικό Σφάλμα (MSE) μεταξύ προβλέψεων και
πραγματικών τιμών.
    - y_pred: Προβλέψεις του δικτύου (Tensor με shape (N, 1))
    - y_true: Πραγματικές τιμές (Tensor με shape (N, 1))
    - Επιστρέφει: MSE loss (Tensor scalar)
    """
    return torch.mean((y_pred - y_true) ** 2)  # Μέση τιμή τετραγωνικών
σφαλμάτων
```

```
# ------------- Μέθοδος Runge-Kutta 4ης τάξης (RK4) για Βελτιστοποίηση -------------

def runge_kutta_optimizer(model, loss_fn, x_batch, y_batch, h=0.1):

    """

     Εφαρμόζει τον αλγόριθμο RK4 για ενημέρωση των παραμέτρων ενός νευρωνικού δικτύου.

    - model: Το νευρωνικό δίκτυο (π.χ. torch.nn.Module)

    - loss_fn: Συνάρτηση κόστους (π.χ. MSE)

    - x_batch: Δεδομένα εισόδου (Tensor)

    - y_batch: Ετικέτες (Tensor)

    - h: Ρυθμός μάθησης (learning rate)

    - Επιστρέφει: Το ενημερωμένο μοντέλο

   Θεωρία: Ο RK4 προσέγγιζει τη λύση διαφορικών εξισώσεων με συνδυασμό 4 κλίσεων (k1-k4).

    """


    # 1. Αποθήκευση αρχικών βαρών ως ένα διανύσματος (για πιο εύκολες πράξεις)

                                        original_params                 = torch.nn.utils.parameters_to_vector(model.parameters()).clone()


    # 2. Βοηθητική συνάρτηση για προσωρινή αλλαγή παραμέτρων

    @contextlib.contextmanager

    def temp_params(temp):

        """Προσωρινά ενημερώνει τις παραμέτρους του μοντέλου και μετά τις επαναφέρει."""

                              torch.nn.utils.vector_to_parameters(temp, model.parameters())  # Εφαρμογή νέων παραμέτρων

        yield  # Παύση για υπολογισμούς

        torch.nn.utils.parameters_to_vector(model.parameters()).copy_(original_params)  # Επαναφορά αρχικών


    # 3. Υπολογισμός 4 βημάτων κλίσης (k1, k2, k3, k4)

    grads = []

    # Διατρέχουμε τα 4 στάδια του RK4: [0, h/2, h/2, h]

    for coeff in [0, h/2, h/2, h]:

        with temp_params(original_params + coeff * (grads[-1] if grads else 0)):  # Προσωρινή ενημέρωση βαρών
```

```
            model.zero_grad()  # Μηδενισμός προηγούμενων κλίσεων
            loss = loss_fn(model(x_batch), y_batch)  # Forward pass
            loss.backward()  # Backpropagation (υπολογισμός κλίσεων)
            # Αποθήκευση κλίσεων ως επίπεδου διανύσματος
             grad = torch.nn.utils.parameters_to_vector([p.grad for p
in model.parameters()])
            grads.append(grad)  # Προσθήκη στη λίστα grad


    # 4. Διαχωρισμός των υπολογισμένων κλίσεων
    k1, k2, k3, k4 = grads  # Ανάθεση σε k1-k4


    # 5. Τελική ενημέρωση βαρών με τον τύπο RK4
    new_params = original_params - (h / 6) * (k1 + 2*k2 + 2*k3 + k4)  #
Συνδυασμός κλίσεων
                      torch.nn.utils.vector_to_parameters(new_params,
model.parameters())  # Εφαρμογή νέων βαρών


    return model


# ------------- Δημιουργία Δεδομένων -------------
t_start = 0.0 # Αρχικός χρόνος
t_end = 1.9  # Τερματίζουμε πριν το t=2 γιατί η συνάρτηση απειρίζεται
εκεί
num_points = 100  # Αριθμός σημείων εκπαίδευσης
x_train = np.linspace(t_start, t_end, num_points).reshape(-1, 1)  #
Διάνυσμα εισόδου t (shape: (100, 1))
y_train = 1 / (np.exp(x_train) * (2 - x_train))  # Αναλυτική λύση y(t)
= 1/(e^t (2 - t))


# Μετατροπή σε PyTorch Tensors (απαιτείται για την εκπαίδευση)
x_train_tensor  =  torch.tensor(x_train,  dtype=torch.float32)  #
Δεδομένα εισόδου
y_train_tensor  =  torch.tensor(y_train,  dtype=torch.float32)  #
Ετικέτες


# ------------- Αρχικοποίηση Μοντέλου -------------
model = SimpleNN()  # Δημιουργία νευρωνικού δικτύου
```

```
# ------------- Εκπαίδευση Μοντέλου -------------
epochs = 2000  # Αριθμός επαναλήψεων εκπαίδευσης
learning_rate = 0.1  # Βήμα εκπαίδευσης (h)
losses = []  # Λίστα για καταγραφή ιστορικού απωλειών

for epoch in range(epochs):
    model.train()  # Ορισμός μοντέλου σε κατάσταση εκπαίδευσης
    # Εκτέλεση βήματος βελτιστοποιητή RK4
     model = runge_kutta_optimizer(model, loss_fn, x_train_tensor, y_train_tensor, h=learning_rate)

    # Υπολογισμός τρέχουσας απώλειας
    with torch.no_grad():  # Απενεργοποίηση υπολογισμού κλίσεων (για ταχύτητα)
        y_pred = model(x_train_tensor)
        loss = loss_fn(y_pred, y_train_tensor)
        losses.append(loss.item())  # Αποθήκευση απώλειας

    # Εκτύπωση ανά 100 εποχές
    if epoch % 100 == 0:
          print(f"Epoch {epoch}, Loss: {loss.item()}")  # Εμφάνιση απώλειας

# ------------- Απεικόνιση Αποτελεσμάτων -------------
# Δημιουργία γραφήματος απώλειας
plt.plot(losses)  # Σχεδίαση ιστορικού απωλειών
plt.title('Εξέλιξη Απώλειας Κατά την Εκπαίδευση')  # Τίτλος γραφήματος
plt.xlabel('Εποχή')  # Ετικέτα άξονα x
plt.ylabel('Απώλεια')  # Ετικέτα άξονα y
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος

# Πρόβλεψη με το εκπαιδευμένο μοντέλο
model.eval()  # Ορισμός μοντέλου σε λειτουργία αξιολόγησης
with torch.no_grad():  # Απενεργοποίηση υπολογισμού gradients για ταχύτητα
```

```
    y_pred = model(x_train_tensor).detach().numpy()  # Μετατροπή προβλέψεων σε numpy array


# Σύγκριση αναλυτικής λύσης και προβλέψεων
plt.plot(x_train, y_train, label='Αναλυτική Λύση: $1/(e^t (2 - t))$')  # Σχεδίαση πραγματικών τιμών
plt.plot(x_train, y_pred, '--', label='Πρόβλεψη Νευρωνικού Δικτύου')  # Σχεδίαση προβλέψεων με διακεκομμένη γραμμή
plt.title('Σύγκριση Αναλυτικής Λύσης και Προβλέψεων')  # Τίτλος γραφήματος
plt.xlabel('t')  # Ετικέτα άξονα x
plt.ylabel('y(t)')  # Ετικέτα άξονα y
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος
```

Υλοποίηση ενός Νευρωνικού Δικτύου για την Προσέγγιση Λύσης ΣΔΕ (με Full-Batch Gradient Descent ως βελτιστοποιητή)

```
# ------------- Εισαγωγή Βιβλιοθηκών -------------
import torch  # Βασική βιβλιοθήκη για δουλειά με tensors και νευρωνικά δίκτυα
import torch.nn as nn  # Επίπεδα νευρωνικών δικτύων (π.χ. Linear, ReLU)
import torch.optim as optim  # Βελτιστοποιητές (Adam, SGD κλπ)
import numpy as np  # Βιβλιοθήκη για επιστημονικούς υπολογισμούς
import matplotlib.pyplot as plt  # Βιβλιοθήκη για δημιουργία γραφημάτων

# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Ορισμός Νευρωνικού Δικτύου -------------
class SimpleNN(nn.Module):
    def __init__(self):
        """
        Αρχικοποιεί ένα δίκτυο με:
```

```
        - fc1: Πρώτο γραμμικό επίπεδο (1 είσοδος -> 10 νευρώνες)
        - fc2: Δεύτερο γραμμικό επίπεδο (10 νευρώνες -> 1 έξοδος)
        """
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(
            in_features=1,     # Διαστάσεις εισόδου
            out_features=10    # Αριθμός νευρώνων στο κρυφό επίπεδο
        )
        self.fc2 = nn.Linear(
            in_features=10,    # Είσοδος από τους 10 νευρώνες του fc1
            out_features=1     # Έξοδος (πρόβλεψη)
        )

    def forward(self, x):
        """
        Forward pass του δικτύου:
        - x: Tensor εισόδου με shape (N, 1), όπου N = αριθμός δειγμάτων
        - Επιστρέφει: Tensor εξόδου με shape (N, 1)
        """
         x = torch.relu(self.fc1(x))  # ReLU ενεργοποίηση στο κρυφό
επίπεδο
        x = self.fc2(x)              # Γραμμικός μετασχηματισμός στην
έξοδο
        return x


# ------------- Συνάρτηση Κόστους (MSE) -------------
def loss_fn(y_pred, y_true):
    """
    Υπολογίζει το Μέσο Τετραγωνικό Σφάλμα (MSE) μεταξύ προβλέψεων και
πραγματικών τιμών.
    - y_pred: Προβλέψεις μοντέλου (Tensor)
    - y_true: Πραγματικές τιμές (Tensor)
    - Επιστρέφει: MSE loss (Tensor scalar)
    """
    return torch.mean((y_pred - y_true)**2)  # Μέση τιμή τετραγωνικών
σφαλμάτων
```

```
# ------------- Δημιουργία Δεδομένων -------------

t_start = 0.0 # Αρχικός χρόνος

t_end = 1.9  # Τερματίζουμε πριν το t=2 γιατί η συνάρτηση απειρίζεται εκεί

num_points = 100  # Αριθμός σημείων εκπαίδευσης

x_train = np.linspace(t_start, t_end, num_points).reshape(-1, 1)  # Διάνυσμα εισόδου t (shape: (100, 1))

y_train = 1 / (np.exp(x_train) * (2 - x_train))  # Αναλυτική λύση y(t) = 1/(e^t (2 - t))


# Μετατροπή σε PyTorch Tensors (απαιτείται για την εκπαίδευση)

x_train_tensor = torch.tensor(x_train, dtype=torch.float32)  # Δεδομένα εισόδου

y_train_tensor = torch.tensor(y_train, dtype=torch.float32)  # Ετικέτες


# ------------- Αρχικοποίηση Μοντέλου & Βελτιστοποιητή -------------

model = SimpleNN()  # Δημιουργία στιγμιότυπου νευρωνικού δικτύου

optimizer = optim.SGD(model.parameters(), lr=0.1)  # Full-Batch GD με ρυθμό μάθησης 0.1

losses = []  # Λίστα για καταγραφή ιστορικού απωλειών


# ------------- Εκπαίδευση Full-Batch -------------

epochs = 2000  # Συνολικός αριθμός εποχών εκπαίδευσης


for epoch in range(epochs):
    model.train()  # Ορισμός μοντέλου σε κατάσταση εκπαίδευσης


    # Forward pass (Χρήση ΟΛΩΝ των δεδομένων)

    pred = model(x_train_tensor)  # Πρόβλεψη για όλο το dataset

    loss = loss_fn(pred, y_train_tensor)  # Υπολογισμός MSE


    # Backward pass

    optimizer.zero_grad()  # Μηδενισμός gradients

    loss.backward()  # Αυτόματος υπολογισμός gradients

    optimizer.step()  # Ενημέρωση βαρών
```

```
    # Αποθήκευση απώλειας
    losses.append(loss.item())  # Προσθήκη τρέχουσας απώλειας

    # Εκτύπωση ανά 100 εποχές
    if epoch % 100 == 0:
        print(f"Epoch {epoch}, Loss: {loss.item()}")  # Εμφάνιση απώλειας

# ------------- Απεικόνιση Αποτελεσμάτων -------------
# Δημιουργία γραφήματος απώλειας
plt.plot(losses)  # Σχεδίαση ιστορικού απωλειών
plt.title('Εξέλιξη Απώλειας Κατά την Εκπαίδευση')  # Τίτλος γραφήματος
plt.xlabel('Εποχή')  # Ετικέτα άξονα x
plt.ylabel('Απώλεια')  # Ετικέτα άξονα y
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος

# Πρόβλεψη με το εκπαιδευμένο μοντέλο
model.eval()  # Ορισμός μοντέλου σε λειτουργία αξιολόγησης
with torch.no_grad():  # Απενεργοποίηση υπολογισμού gradients για ταχύτητα
    y_pred = model(x_train_tensor).detach().numpy()  # Μετατροπή προβλέψεων σε numpy array

# Σύγκριση αναλυτικής λύσης και προβλέψεων
plt.plot(x_train, y_train, label='Αναλυτική Λύση: $1/(e^t (2 - t))$')  # Σχεδίαση πραγματικών τιμών
plt.plot(x_train, y_pred, '--', label='Πρόβλεψη Νευρωνικού Δικτύου')  # Σχεδίαση προβλέψεων με διακεκομμένη γραμμή
plt.title('Σύγκριση Αναλυτικής Λύσης και Προβλέψεων')  # Τίτλος γραφήματος
plt.xlabel('t')  # Ετικέτα άξονα x
plt.ylabel('y(t)')  # Ετικέτα άξονα y
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος
```

Υλοποίηση ενός Νευρωνικού Δικτύου για την Προσέγγιση Λύσης ΣΔΕ (με τον βελτιστοποιητή Adam)

```
# ------------- Εισαγωγή Βιβλιοθηκών -------------
import torch  # Βασική βιβλιοθήκη για δουλειά με tensors και νευρωνικά δίκτυα
import torch.nn as nn  # Επίπεδα νευρωνικών δικτύων (π.χ. Linear, ReLU)
import torch.optim as optim  # Βελτιστοποιητές (Adam, SGD κλπ)
import numpy as np  # Βιβλιοθήκη για επιστημονικούς υπολογισμούς
import matplotlib.pyplot as plt  # Βιβλιοθήκη για δημιουργία γραφημάτων

# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Ορισμός Νευρωνικού Δικτύου -------------
class SimpleNN(nn.Module):
    def __init__(self):
        """
        Αρχικοποιεί ένα δίκτυο με:
        - fc1: Πρώτο γραμμικό επίπεδο (1 είσοδος -> 10 νευρώνες)
        - fc2: Δεύτερο γραμμικό επίπεδο (10 νευρώνες -> 1 έξοδος)
        """
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(
            in_features=1,     # Διαστάσεις εισόδου
            out_features=10    # Αριθμός νευρώνων στο κρυφό επίπεδο
        )
        self.fc2 = nn.Linear(
            in_features=10,    # Είσοδος από τους 10 νευρώνες του fc1
            out_features=1     # Έξοδος (πρόβλεψη)
        )

    def forward(self, x):
```

```
        """
        Forward pass του δικτύου:
        - x: Tensor εισόδου με shape (N, 1), όπου N = αριθμός δειγμάτων
        - Επιστρέφει: Tensor εξόδου με shape (N, 1)
        """
         x = torch.relu(self.fc1(x))  # ReLU ενεργοποίηση στο κρυφό επίπεδο
        x = self.fc2(x)              # Γραμμικός μετασχηματισμός στην έξοδο
        return x


# ------------- Συνάρτηση Κόστους (MSE) -------------
def loss_fn(y_pred, y_true):
    """
    Υπολογίζει το Μέσο Τετραγωνικό Σφάλμα (MSE) μεταξύ προβλέψεων και πραγματικών τιμών.
    - y_pred: Προβλέψεις του δικτύου (Tensor με shape (N, 1))
    - y_true: Πραγματικές τιμές (Tensor με shape (N, 1))
    - Επιστρέφει: MSE loss (Tensor scalar)
    """
    return torch.mean((y_pred - y_true) ** 2)  # Μέση τιμή τετραγωνικών σφαλμάτων


# ------------- Δημιουργία Δεδομένων -------------
t_start = 0.0 # Αρχικός χρόνος
t_end = 1.9  # Τερματίζουμε πριν το t=2 γιατί η συνάρτηση απειρίζεται εκεί
num_points = 100  # Αριθμός σημείων εκπαίδευσης
x_train = np.linspace(t_start, t_end, num_points).reshape(-1, 1)  # Διάνυσμα εισόδου t (shape: (100, 1))
y_train = 1 / (np.exp(x_train) * (2 - x_train))  # Αναλυτική λύση y(t) = 1/(e^t (2 - t))


# Μετατροπή σε PyTorch Tensors (απαιτείται για την εκπαίδευση)
x_train_tensor = torch.tensor(x_train, dtype=torch.float32)  # Δεδομένα εισόδου
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)  # Ετικέτες
```

```
# ------------- Αρχικοποίηση Μοντέλου & Βελτιστοποιητή -------------
model = SimpleNN()  # Δημιουργία στιγμιότυπου του νευρωνικού δικτύου
optimizer = optim.Adam(model.parameters(), lr=0.01)  # Βελτιστοποιητής Adam με ρυθμό μάθησης 0.01
losses = []  # Λίστα για καταγραφή ιστορικού απωλειών

# ------------- Εκπαίδευση Μοντέλου -------------
epochs = 2000  # Συνολικός αριθμός εποχών εκπαίδευσης

for epoch in range(epochs):
    model.train()  # Ορισμός μοντέλου σε κατάσταση εκπαίδευσης
    optimizer.zero_grad()  # Μηδενισμός gradients από προηγούμενη εποχή

    # Forward pass
    y_pred = model(x_train_tensor)  # Πρόβλεψη εξόδων από το δίκτυο

    # Υπολογισμός απώλειας
    loss = loss_fn(y_pred, y_train_tensor)  # Υπολογισμός MSE

    # Backward pass
     loss.backward()  # Αυτόματος υπολογισμός gradients (αλυσιδωτός κανόνας)

    # Ενημέρωση βαρών
    optimizer.step()  # Εφαρμογή βημάτων βελτιστοποίησης

    # Καταγραφή απώλειας
    losses.append(loss.item())  # Αποθήκευση τρέχουσας απώλειας

    # Εκτύπωση ανά 100 εποχές
    if epoch % 100 == 0:
          print(f"Epoch {epoch}, Loss: {loss.item()}")  # Εμφάνιση απώλειας

# ------------- Απεικόνιση Αποτελεσμάτων -------------
```

```
# Δημιουργία γραφήματος απώλειας
plt.plot(losses)  # Σχεδίαση ιστορικού απωλειών
plt.title('Εξέλιξη Απώλειας Κατά την Εκπαίδευση')  # Τίτλος γραφήματος
plt.xlabel('Εποχή')  # Ετικέτα άξονα x
plt.ylabel('Απώλεια')  # Ετικέτα άξονα y
plt.grid(True)  # Προσθήκη πλέγματος
plt.show()  # Εμφάνιση γραφήματος

# Πρόβλεψη με το εκπαιδευμένο μοντέλο
model.eval()  # Ορισμός μοντέλου σε λειτουργία αξιολόγησης
with torch.no_grad():  # Απενεργοποίηση υπολογισμού gradients για ταχύτητα
    y_pred = model(x_train_tensor).detach().numpy()  # Μετατροπή προβλέψεων σε numpy array

# Σύγκριση αναλυτικής λύσης και προβλέψεων
plt.plot(x_train, y_train, label='Αναλυτική Λύση: $1/(e^t (2 - t))$')  # Σχεδίαση πραγματικών τιμών
plt.plot(x_train, y_pred, '--', label='Πρόβλεψη Νευρωνικού Δικτύου')  # Σχεδίαση προβλέψεων με διακεκομμένη γραμμή
plt.title('Σύγκριση Αναλυτικής Λύσης και Προβλέψεων')  # Τίτλος γραφήματος
plt.xlabel('t')  # Ετικέτα άξονα x
plt.ylabel('y(t)')  # Ετικέτα άξονα y
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος
plt.show()  # Εμφάνιση γραφήματος
```

Σύγκριση Πολλαπλών Βελτιστοποιητών (custom RK4,Full-Batch GD & Adam) για την Προσέγγιση Λύσης ΣΔΕ με Νευρωνικά Δίκτυα

```
# ------------- Εισαγωγή Βιβλιοθηκών -------------
import torch  # Βασική βιβλιοθήκη για δουλειά με tensors και νευρωνικά δίκτυα
import torch.nn as nn  # Επίπεδα νευρωνικών δικτύων (π.χ. Linear, ReLU)
```

```
import torch.optim as optim  # Βελτιστοποιητές (Adam, SGD κλπ)
import numpy as np  # Βιβλιοθήκη για επιστημονικούς υπολογισμούς
import matplotlib.pyplot as plt  # Βιβλιοθήκη για δημιουργία γραφημάτων
import contextlib # Βιβλιοθήκη για τη διαχείριση context managers

# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Ορισμός Νευρωνικού Δικτύου -------------
class SimpleNN(nn.Module):
    def __init__(self):
        """
        Αρχικοποιεί ένα δίκτυο με:
        - fc1: Πρώτο γραμμικό επίπεδο (1 είσοδος -> 10 νευρώνες)
        - fc2: Δεύτερο γραμμικό επίπεδο (10 νευρώνες -> 1 έξοδος)
        """
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(
            in_features=1,     # Διαστάσεις εισόδου
            out_features=10    # Αριθμός νευρώνων στο κρυφό επίπεδο
        )
        self.fc2 = nn.Linear(
            in_features=10,    # Είσοδος από τους 10 νευρώνες του fc1
            out_features=1     # Έξοδος (πρόβλεψη)
        )

    def forward(self, x):
        """
        Forward pass του δικτύου:
        - x: Tensor εισόδου με shape (N, 1), όπου N = αριθμός δειγμάτων
        - Επιστρέφει: Tensor εξόδου με shape (N, 1)
        """
        x = torch.relu(self.fc1(x))   # ReLU ενεργοποίηση στο κρυφό
επίπεδο
```

```
        x = self.fc2(x)              # Γραμμικός μετασχηματισμός στην
έξοδο
        return x


# ------------- Συνάρτηση Κόστους (MSE) -------------
def loss_fn(y_pred, y_true):
    """
    Υπολογίζει το Μέσο Τετραγωνικό Σφάλμα (MSE) μεταξύ προβλέψεων και
πραγματικών τιμών.
    - y_pred: Προβλέψεις του δικτύου (Tensor με shape (N, 1))
    - y_true: Πραγματικές τιμές (Tensor με shape (N, 1))
    - Επιστρέφει: MSE loss (Tensor scalar)
    """
    return torch.mean((y_pred - y_true) ** 2)  # Μέση τιμή τετραγωνικών
σφαλμάτων


# ------------- Μέθοδος Runge-Kutta 4ης τάξης (RK4) για Βελτιστοποίηση
-------------
def runge_kutta_optimizer(model, loss_fn, x_batch, y_batch, h=0.1):
    """
    Εφαρμόζει τον αλγόριθμο RK4 για ενημέρωση των παραμέτρων ενός
νευρωνικού δικτύου.
    - model: Το νευρωνικό δίκτυο (π.χ. torch.nn.Module)
    - loss_fn: Συνάρτηση κόστους (π.χ. MSE)
    - x_batch: Δεδομένα εισόδου (Tensor)
    - y_batch: Ετικέτες (Tensor)
    - h: Ρυθμός μάθησης (learning rate)
    - Επιστρέφει: Το ενημερωμένο μοντέλο
    Θεωρία: Ο RK4 προσέγγιζει τη λύση διαφορικών εξισώσεων με συνδυασμό
4 κλίσεων (k1-k4).
    """

    # 1. Αποθήκευση αρχικών βαρών ως ένα διανύσματος (για πιο εύκολες
πράξεις)
    original_params                                                =
torch.nn.utils.parameters_to_vector(model.parameters()).clone()

    # 2. Βοηθητική συνάρτηση για προσωρινή αλλαγή παραμέτρων
```

```
    @contextlib.contextmanager
    def temp_params(temp):
        """Προσωρινά ενημερώνει τις παραμέτρους του μοντέλου και μετά τις επαναφέρει."""
        torch.nn.utils.vector_to_parameters(temp, model.parameters()) # Εφαρμογή νέων παραμέτρων
        yield  # Παύση για υπολογισμούς

torch.nn.utils.parameters_to_vector(model.parameters()).copy_(original_params)  # Επαναφορά αρχικών


    # 3. Υπολογισμός 4 βημάτων κλίσης (k1, k2, k3, k4)
    grads = []
    # Διατρέχουμε τα 4 στάδια του RK4: [0, h/2, h/2, h]
    for coeff in [0, h/2, h/2, h]:
        with temp_params(original_params + coeff * (grads[-1] if grads else 0)):  # Προσωρινή ενημέρωση βαρών
            model.zero_grad()  # Μηδενισμός προηγούμενων κλίσεων
            loss = loss_fn(model(x_batch), y_batch)  # Forward pass
            loss.backward()  # Backpropagation (υπολογισμός κλίσεων)
            # Αποθήκευση κλίσεων ως επίπεδου διανύσματος
            grad = torch.nn.utils.parameters_to_vector([p.grad for p in model.parameters()])
            grads.append(grad)  # Προσθήκη στη λίστα grad


    # 4. Διαχωρισμός των υπολογισμένων κλίσεων
    k1, k2, k3, k4 = grads  # Ανάθεση σε k1-k4


    # 5. Τελική ενημέρωση βαρών με τον τύπο RK4
    new_params = original_params - (h / 6) * (k1 + 2*k2 + 2*k3 + k4) # Συνδυασμός κλίσεων
    torch.nn.utils.vector_to_parameters(new_params, model.parameters())  # Εφαρμογή νέων βαρών


    return model


# ------------- Δημιουργία Δεδομένων -------------
t_start = 0.0 # Αρχικός χρόνος
```

```
t_end = 1.9  # Τερματίζουμε πριν το t=2 γιατί η συνάρτηση απειρίζεται εκεί

num_points = 100  # Αριθμός σημείων εκπαίδευσης

x_train = np.linspace(t_start, t_end, num_points).reshape(-1, 1)  # Διάνυσμα εισόδου t (shape: (100, 1))

y_train = 1 / (np.exp(x_train) * (2 - x_train))  # Αναλυτική λύση y(t) = 1/(e^t (2 - t))


# Μετατροπή σε PyTorch Tensors (απαιτείται για την εκπαίδευση)

x_train_tensor = torch.tensor(x_train, dtype=torch.float32)  # Δεδομένα εισόδου

y_train_tensor = torch.tensor(y_train, dtype=torch.float32)  # Ετικέτες


# ------------- Παραμέτρους Εκπαίδευσης -------------

EPOCHS = 2000  # Αριθμός επαναλήψεων εκπαίδευσης

LR = 0.1  # Κοινό learning rate για όλους τους βελτιστοποιητές


# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------

torch.manual_seed(42)  # Σταθερό seed για PyTorch

np.random.seed(42)  # Σταθερό seed για NumPy


# ------------- Εκπαίδευση με RK4 -------------

model_rk4 = SimpleNN()  # Αρχικοποίηση μοντέλου

rk4_losses = []  # Λίστα για την καταγραφή των απωλειών

print("\n--- Εκπαίδευση με RK4 ---")

for epoch in range(EPOCHS):

    model_rk4 = runge_kutta_optimizer(model_rk4, loss_fn, x_train_tensor, y_train_tensor, h=LR)

    with torch.no_grad():  # Απενεργοποίηση υπολογισμού gradients για ταχύτητα

        loss = loss_fn(model_rk4(x_train_tensor), y_train_tensor)

        rk4_losses.append(loss.item())  # Αποθήκευση τρέχουσας απώλειας

    if epoch % 100 == 0:  # Εκτύπωση ανά 100 εποχές

        print(f"Epoch {epoch}: Loss = {loss.item()}")  # Εμφάνιση απώλειας
```

```
# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Εκπαίδευση με Full-Batch GD -------------
model_sgd = SimpleNN()
optimizer_sgd = optim.SGD(model_sgd.parameters(), lr=LR)  # GD με Full-
Batch
sgd_losses = []
print("\n--- Εκπαίδευση με Full-Batch GD ---")
for epoch in range(EPOCHS):
    model_sgd.train()
    pred = model_sgd(x_train_tensor)    # Πρόβλεψη για όλο το dataset
    loss = loss_fn(pred, y_train_tensor)  # Υπολογισμός απώλειας
    optimizer_sgd.zero_grad()  # Μηδενισμός gradients
    loss.backward()  # Backpropagation
    optimizer_sgd.step()  # Ενημέρωση βαρών
    sgd_losses.append(loss.item())  # Αποθήκευση τρέχουσας απώλειας
    if epoch % 100 == 0:  # Εκτύπωση ανά 100 εποχές
        print(f"Epoch {epoch}: Loss = {loss.item()}")

# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Εκπαίδευση με Adam -------------
model_adam = SimpleNN()  # Αρχικοποίηση μοντέλου
optimizer_adam  =  optim.Adam(model_adam.parameters(),  lr=LR)    #
Βελτιστοποιητής Adam
adam_losses = []  # Λίστα για την καταγραφή των απωλειών
print("\n--- Εκπαίδευση με Adam ---")
for epoch in range(EPOCHS):
    optimizer_adam.zero_grad()  # Μηδενισμός gradients
    pred = model_adam(x_train_tensor)  # Πρόβλεψη
    loss = loss_fn(pred, y_train_tensor)  # Υπολογισμός απώλειας
```

```
    loss.backward()  # Backpropagation
    optimizer_adam.step()  # Ενημέρωση βαρών
    adam_losses.append(loss.item())  # Αποθήκευση τρέχουσας απώλειας
    if epoch % 100 == 0:  # Εκτύπωση ανά 100 εποχές
        print(f"Epoch {epoch}: Loss = {loss.item()}")

# ------------- Πρόβλεψεις -------------
def get_predictions(model):
    """Επιστρέφει τις προβλέψεις του μοντέλου σε numpy array"""
    model.eval()  # Ορισμός μοντέλου σε λειτουργία αξιολόγησης
    with torch.no_grad():  # Απενεργοποίηση υπολογισμού gradients
        return model(x_train_tensor).numpy()

y_pred_rk4 = get_predictions(model_rk4)  # Προβλέψεις RK4
y_pred_sgd = get_predictions(model_sgd)  # Προβλέψεις Full-Batch GD
y_pred_adam = get_predictions(model_adam)  # Προβλέψεις Adam

# ------------- Γραφήματα -------------
plt.figure(figsize=(14, 10))   # Δημιουργία figure με μέγεθος 14x10 ίντσες

# 1. Σύγκριση Απωλειών
plt.subplot(2, 2, 1)
plt.plot(rk4_losses, label='RK4')  # Σχεδίαση απωλειών RK4
plt.plot(sgd_losses, label='Full-Batch GD')  # Σχεδίαση απωλειών GD
plt.plot(adam_losses, label='Adam')  # Σχεδίαση απωλειών Adam
plt.title('Σύγκριση Απωλειών')  # Τίτλος γραφήματος
plt.xlabel('Εποχή')  # Ετικέτα άξονα x
plt.ylabel('MSE')  # Ετικέτα άξονα y
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος

# 2. Σύγκριση Προβλέψεων
plt.subplot(2, 2, 2)
```

```
plt.plot(x_train, y_train, 'k-', linewidth=2, label='Αναλυτική')  # Σχεδίαση αναλυτικής λύσης

plt.plot(x_train, y_pred_rk4, label='RK4')  # Σχεδίαση προβλέψεων RK4

plt.plot(x_train, y_pred_sgd, label='Full-Batch GD')  # Σχεδίαση προβλέψεων GD

plt.plot(x_train, y_pred_adam, label='Adam')  # Σχεδίαση προβλέψεων Adam

plt.title('Σύγκριση Προβλέψεων')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('y(t)')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος


# 3. Απόλυτο Σφάλμα (Γραμμική Κλίμακα)

plt.subplot(2, 2, 3)

plt.plot(x_train, np.abs(y_pred_rk4 - y_train), label='RK4')  # Σφάλμα RK4

plt.plot(x_train, np.abs(y_pred_sgd - y_train), label='Full-Batch GD')  # Σφάλμα GD

plt.plot(x_train, np.abs(y_pred_adam - y_train), label='Adam')  # Σφάλμα Adam

plt.title('Απόλυτο Σφάλμα (Γραμμική Κλίμακα)')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('|Δy|')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος


# 4. Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)

plt.subplot(2, 2, 4)

plt.semilogy(x_train, np.abs(y_pred_rk4 - y_train), label='RK4')  # Σφάλμα RK4

plt.semilogy(x_train, np.abs(y_pred_sgd - y_train), label='Full-Batch GD')  # Σφάλμα GD

plt.semilogy(x_train, np.abs(y_pred_adam - y_train), label='Adam')  # Σφάλμα Adam

plt.title('Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('|Δy|')  # Ετικέτα άξονα y
```

```
plt.legend()  # Εμφάνιση λεζάντας
plt.grid(True)  # Εμφάνιση πλέγματος

plt.tight_layout()  # Βελτιστοποίηση διάταξης υπογραφημάτων
plt.show()  # Εμφάνιση όλων των γραφημάτων
```

Σύγκριση Πολλαπλών Βελτιστοποιητών (custom RK4,Full-Batch GD & Adam) για την Προσέγγιση Λύσης ΣΔΕ με Νευρωνικά Δίκτυα (με τα πιθανά καλύτερα h/lr για κάθε βελτιστοποιητή)

```
# ------------- Εισαγωγή Βιβλιοθηκών -------------
import torch  # Βασική βιβλιοθήκη για δουλειά με tensors και νευρωνικά δίκτυα
import torch.nn as nn  # Επίπεδα νευρωνικών δικτύων (π.χ. Linear, ReLU)
import torch.optim as optim  # Βελτιστοποιητές (Adam, SGD κλπ)
import numpy as np  # Βιβλιοθήκη για επιστημονικούς υπολογισμούς
import matplotlib.pyplot as plt  # Βιβλιοθήκη για δημιουργία γραφημάτων
import contextlib # Βιβλιοθήκη για τη διαχείριση context managers

# ------------- Ορισμός Μοντέλου -------------
class SimpleNN(nn.Module):
    def __init__(self):
        """
        Αρχικοποιεί ένα δίκτυο με:
        - fc1: Πρώτο γραμμικό επίπεδο (1 είσοδος -> 10 νευρώνες)
        - fc2: Δεύτερο γραμμικό επίπεδο (10 νευρώνες -> 1 έξοδος)
        """
        super().__init__()
        self.fc1 = nn.Linear(1, 10)  # Γραμμικό επίπεδο εισόδου-κρυφού
        self.fc2 = nn.Linear(10, 1)  # Γραμμικό επίπεδο κρυφού-εξόδου

    def forward(self, x):
        """Forward pass του δικτύου"""
        x = torch.relu(self.fc1(x))  # Ενεργοποίηση ReLU στο κρυφό επίπεδο
```

```
        return self.fc2(x)  # Γραμμικός μετασχηματισμός εξόδου


# ------------- Συνάρτηση Κόστους -------------

def loss_fn(y_pred, y_true):

    """

    Υπολογίζει το Μέσο Τετραγωνικό Σφάλμα (MSE)

    - y_pred: Προβλέψεις μοντέλου (Tensor)

    - y_true: Πραγματικές τιμές (Tensor)

    """

    return torch.mean((y_pred - y_true)**2)  # Τύπος MSE


# ------------- Μέθοδος Runge-Kutta 4ης τάξης (RK4) για Βελτιστοποίηση -------------

def runge_kutta_optimizer(model, loss_fn, x_batch, y_batch, h=0.1):

    """

    Εφαρμόζει τον αλγόριθμο RK4 για ενημέρωση των παραμέτρων ενός νευρωνικού δικτύου.

    - model: Το νευρωνικό δίκτυο (π.χ. torch.nn.Module)

    - loss_fn: Συνάρτηση κόστους (π.χ. MSE)

    - x_batch: Δεδομένα εισόδου (Tensor)

    - y_batch: Ετικέτες (Tensor)

    - h: Ρυθμός μάθησης (learning rate)

    - Επιστρέφει: Το ενημερωμένο μοντέλο

    Θεωρία: Ο RK4 προσέγγιζει τη λύση διαφορικών εξισώσεων με συνδυασμό 4 κλίσεων (k1-k4).

    """


    # 1. Αποθήκευση αρχικών βαρών ως ένα διανύσματος (για πιο εύκολες πράξεις)

    original_params = torch.nn.utils.parameters_to_vector(model.parameters()).clone()


    # 2. Βοηθητική συνάρτηση για προσωρινή αλλαγή παραμέτρων

    @contextlib.contextmanager

    def temp_params(temp):

        """Προσωρινά ενημερώνει τις παραμέτρους του μοντέλου και μετά τις επαναφέρει."""
```

```
        torch.nn.utils.vector_to_parameters(temp, model.parameters())
# Εφαρμογή νέων παραμέτρων
        yield  # Παύση για υπολογισμούς

torch.nn.utils.parameters_to_vector(model.parameters()).copy_(origina
l_params)  # Επαναφορά αρχικών


    # 3. Υπολογισμός 4 βημάτων κλίσης (k1, k2, k3, k4)
    grads = []
    # Διατρέχουμε τα 4 στάδια του RK4: [0, h/2, h/2, h]
    for coeff in [0, h/2, h/2, h]:
        with temp_params(original_params + coeff * (grads[-1] if grads
else 0)):  # Προσωρινή ενημέρωση βαρών
            model.zero_grad()  # Μηδενισμός προηγούμενων κλίσεων
            loss = loss_fn(model(x_batch), y_batch)  # Forward pass
            loss.backward()  # Backpropagation (υπολογισμός κλίσεων)
            # Αποθήκευση κλίσεων ως επίπεδου διανύσματος
            grad = torch.nn.utils.parameters_to_vector([p.grad for p
in model.parameters()])
            grads.append(grad)  # Προσθήκη στη λίστα grad


    # 4. Διαχωρισμός των υπολογισμένων κλίσεων
    k1, k2, k3, k4 = grads  # Ανάθεση σε k1-k4


    # 5. Τελική ενημέρωση βαρών με τον τύπο RK4
    new_params = original_params - (h / 6) * (k1 + 2*k2 + 2*k3 + k4)
# Συνδυασμός κλίσεων
    torch.nn.utils.vector_to_parameters(new_params,
model.parameters())  # Εφαρμογή νέων βαρών


    return model


# ------------- Δεδομένα -------------
t = np.linspace(0, 1.9, 100).reshape(-1, 1)  # Δημιουργία χρονικού
άξονα [0, 1.9]
y_analytical = 1/(np.exp(t)*(2 - t))  # Αναλυτική λύση y(t)
x_tensor = torch.tensor(t, dtype=torch.float32)  # Μετατροπή σε Tensor
```

```
y_tensor = torch.tensor(y_analytical, dtype=torch.float32)  # Μετατροπή σε Tensor


# ==========================================================================
# ======================== GRID SEARCH ================================
# ==========================================================================


# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy


# Πιθανές τιμές για grid search
h_values = [0.01, 0.05, 0.2, 0.3, 0.4, 0.5]  # Τιμές βήματος (h) για τον βελτιστοποιητή RK4
lr_values = [0.0001, 0.001, 0.005, 0.01, 0.05, 0.5]  # Τιμές ρυθμού μάθησης για SGD και Adam


# ------------- Grid Search RK4 -------------
def grid_search_rk4():
    """Αναζήτηση βέλτιστου βήματος (h) για τον βελτιστοποιητή RK4"""
    best = {'h': None, 'loss': float('inf'), 'model_state': None}  # Αρχικοποίηση λεξικού για αποθήκευση βέλτιστων παραμέτρων
    for h in h_values:  # Διατρέχουμε όλες τις πιθανές τιμές του βήματος h
        model = SimpleNN()  # Δημιουργούμε νέο μοντέλο για κάθε τιμή h (για ανεξάρτητα πειράματα)
        losses = []  # Λίστα για την καταγραφή της απώλειας ανά εποχή
        for epoch in range(2000):  # Εκπαίδευση για 2000 εποχές
            # Ενημέρωση βαρών με τον προσαρμοσμένο βελτιστοποιητή RK4
            model = runge_kutta_optimizer(model, loss_fn, x_tensor, y_tensor, h)
            # Υπολογισμός απώλειας χωρίς υπολογισμό gradients (για αποδοτικότητα)
            with torch.no_grad():  # Απενεργοποιεί τον autograd engine για γρηγορότερους υπολογισμούς
                loss = loss_fn(model(x_tensor), y_tensor).item()  # Υπολογισμός MSE
```

```
            losses.append(loss)  # Αποθήκευση τρέχουσας απώλειας
        final_loss = losses[-1]  # Απώλεια στην τελευταία εποχή
        if final_loss < best['loss']:  # Έλεγχος αν είναι η καλύτερη συνολική απόδοση
            best = {'h': h, 'loss': final_loss, 'losses': losses, 'model_state': model.state_dict()}  # Ενημέρωση βέλτιστων
        print(f"h={h} | Τελικό loss: {final_loss}")  # Εκτύπωση προόδου
    return best  # Επιστροφή βέλτιστων παραμέτρων


# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy


# ------------- Grid Search Full-Batch GD -------------
def grid_search_sgd():
    """Αναζήτηση βέλτιστου learning rate για SGD"""
    best = {'lr': None, 'loss': float('inf'), 'model_state': None}  # Λεξικό για βέλτιστα αποτελέσματα SGD
    for lr in lr_values:  # Δοκιμή όλων των ρυθμών μάθησης
        model = SimpleNN()  # Δημιουργία νέου μοντέλου για κάθε lr
        optimizer = optim.SGD(model.parameters(), lr=lr)  # Αρχικοποίηση SGD optimizer με τρέχον lr
        losses = []  # Λίστα για ιστορικό απωλειών
        for epoch in range(2000):  # Επαναλήψεις εκπαίδευσης
            optimizer.zero_grad()  # Μηδενισμός gradients από προηγούμενη εποχή
            loss = loss_fn(model(x_tensor), y_tensor)  # Forward pass και υπολογισμός απώλειας
            loss.backward()  # Backpropagation (υπολογισμός gradients)
            optimizer.step()  # Ενημέρωση βαρών με βάση τα gradients
            losses.append(loss.item())  # Αποθήκευση απώλειας
        final_loss = losses[-1]  # Τελική απώλεια
        if final_loss < best['loss']:  # Έλεγχος για νέο βέλτιστο
            best = {'lr': lr, 'loss': final_loss, 'losses': losses, 'model_state': model.state_dict()}  # Ενημέρωση
        print(f"lr={lr} | Τελικό loss: {final_loss}")  # Εκτύπωση αποτελέσματος
    return best  # Επιστροφή βέλτιστου lr
```

```
# ------------- Διορθώσεις για Αναπαραγωγιμότητα -------------
torch.manual_seed(42)  # Σταθερό seed για PyTorch
np.random.seed(42)  # Σταθερό seed για NumPy

# ------------- Grid Search Adam -------------
def grid_search_adam():
    """Αναζήτηση βέλτιστου learning rate για Adam"""
    best = {'lr': None, 'loss': float('inf'), 'model_state': None}  # Λεξικό για βέλτιστα αποτελέσματα Adam
    for lr in lr_values:  # Δοκιμή όλων των ρυθμών μάθησης
        model = SimpleNN()  # Καινούριο μοντέλο για κάθε lr
        optimizer = optim.Adam(model.parameters(), lr=lr)  # Αρχικοποίηση Adam optimizer με τρέχον lr
        losses = []  # Ιστορικό απωλειών
        for epoch in range(2000):  # Εκπαίδευση
            optimizer.zero_grad()  # Καθαρισμός gradients
            loss = loss_fn(model(x_tensor), y_tensor)  # Υπολογισμός απώλειας
            loss.backward()  # Backward pass
            optimizer.step()  # Ενημέρωση βαρών (με adaptive learning rates)
            losses.append(loss.item())  # Καταγραφή απώλειας
        final_loss = losses[-1]  # Τελικό MSE
        if final_loss < best['loss']:  # Έλεγχος βελτίωσης
            best = {'lr': lr, 'loss': final_loss, 'losses': losses, 'model_state': model.state_dict()}  # Αποθήκευση
        print(f"lr={lr} | Τελικό loss: {final_loss}")  # Output
    return best  # Επιστροφή βέλτιστων

#
# ======================== ΕΚΤΕΛΕΣΗ & ΑΠΟΤΕΛΕΣΜΑΤΑ ==================
#

# Εκτέλεση grid search
```

```
print("\n=== Grid Search RK4 ===")
rk4_best = grid_search_rk4()  # Αναζήτηση βέλτιστου h για RK4

print("\n=== Grid Search Full-Batch GD ===")
sgd_best = grid_search_sgd()  # Αναζήτηση βέλτιστου lr για Full-Batch
GD

print("\n=== Grid Search Adam ===")
adam_best = grid_search_adam()  # Αναζήτηση βέλτιστου lr για Adam

# Φόρτωμα βέλτιστων μοντέλων
def load_best_model(best_config):
    """Φόρτωμα βέλτιστου μοντέλου από αποθηκευμένα βάρη"""
    model = SimpleNN()
    model.load_state_dict(best_config['model_state'])
    return model

rk4_model = load_best_model(rk4_best)  # Φόρτωμα βέλτιστου RK4
sgd_model = load_best_model(sgd_best)  # Φόρτωμα βέλτιστου Full-Batch
GD
adam_model = load_best_model(adam_best)  # Φόρτωμα βέλτιστου Adam

#
==========================================================================
#                  ========================                  ΓΡΑΦΗΜΑΤΑ
==================================
#
==========================================================================

# Πρόβλεψεις
def get_preds(model):
    """Παραγωγή προβλέψεων από το μοντέλο"""
    model.eval()  # Ορισμός μοντέλου σε evaluation mode
    with torch.no_grad():  # Απενεργοποίηση υπολογισμού gradients
        return model(x_tensor).numpy()  # Μετατροπή σε numpy array
```

```
plt.figure(figsize=(20, 12))  # Δημιουργία figure μεγέθους 20x12 ίντσες


# 1. Σύγκριση Απωλειών

plt.subplot(2, 2, 1)

plt.plot(rk4_best['losses'],  label=f'RK4  (h={rk4_best["h"]})')    #
Σχεδίαση απωλειών RK4

plt.plot(sgd_best['losses'],          label=f'Full-Batch          GD
(lr={sgd_best["lr"]})')  # Σχεδίαση απωλειών Full-Batch GD

plt.plot(adam_best['losses'], label=f'Adam (lr={adam_best["lr"]})')  #
Σχεδίαση απωλειών Adam

plt.title('Εξέλιξη Απώλειας')  # Τίτλος γραφήματος

plt.xlabel('Εποχή')  # Ετικέτα άξονα x

plt.ylabel('MSE')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος


# 2. Σύγκριση Προβλέψεων

plt.subplot(2, 2, 2)

plt.plot(t, y_analytical, 'k-', linewidth=2, label='Αναλυτική Λύση')
# Σχεδίαση αναλυτικής λύσης

plt.plot(t, get_preds(rk4_model), label='RK4')  # Σχεδίαση προβλέψεων
RK4

plt.plot(t, get_preds(sgd_model), label='Full-Batch GD')  # Σχεδίαση
προβλέψεων Full-Batch GD

plt.plot(t,   get_preds(adam_model),   label='Adam')      #   Σχεδίαση
προβλέψεων Adam

plt.title('Σύγκριση Προβλέψεων')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('y(t)')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος


# 3. Απόλυτο Σφάλμα (Γραμμική Κλίμακα)

plt.subplot(2, 2, 3)

plt.plot(t, np.abs(get_preds(rk4_model) - y_analytical), label='RK4')
# Σφάλμα RK4

plt.plot(t, np.abs(get_preds(sgd_model) - y_analytical), label='Full-
Batch GD')  # Σφάλμα Full-Batch GD
```

```
plt.plot(t,     np.abs(get_preds(adam_model)     -     y_analytical),
label='Adam')  # Σφάλμα Adam

plt.title('Απόλυτο Σφάλμα (Γραμμική Κλίμακα)')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('|Δy|')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος


# 4. Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)

plt.subplot(2, 2, 4)

plt.semilogy(t,    np.abs(get_preds(rk4_model)    -    y_analytical),
label='RK4')  # Σφάλμα RK4

plt.semilogy(t,    np.abs(get_preds(sgd_model)    -    y_analytical),
label='Full-Batch GD')  # Σφάλμα Full-Batch GD

plt.semilogy(t,    np.abs(get_preds(adam_model)    -    y_analytical),
label='Adam')  # Σφάλμα Adam

plt.title('Απόλυτο Σφάλμα (Λογαριθμική Κλίμακα)')  # Τίτλος γραφήματος

plt.xlabel('t')  # Ετικέτα άξονα x

plt.ylabel('|Δy|')  # Ετικέτα άξονα y

plt.legend()  # Εμφάνιση λεζάντας

plt.grid(True)  # Εμφάνιση πλέγματος


plt.tight_layout()  # Βελτιστοποίηση διάταξης υπογραφημάτων

plt.show()  # Εμφάνιση όλων των γραφημάτων
```

Βιβλιογραφία

[1] Wang, Y. J., & Lin, C. T. (1998). Runge–Kutta Neural Network for Identification of Dynamical Systems in High Accuracy. IEEE Transactions on Neural Networks, 9(2), 294-307. https://doi.org/10.1109/72.661124

[2] Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach (3rd ed.). Pearson.

[3] Konan, K. J.-C. (2022). A Comprehensive Overview of Artificial Intelligence. Conference Paper presented at AIAA, DPPR, DSA, ICSS, IOTE, NLPTA, WEST, CACIT, BMLI-2022 (pp. 173-194). https://doi.org/10.5121/csit.2022.122314

[4] Brundage, M. (2015). Taking superintelligence seriously [Review of the book Superintelligence: Paths, dangers, strategies, by N. Bostrom]. Futures, 72, 32-35. https://doi.org/10.1016/j.futures.2015.07.009

[5] Gohel, P., Singh, P., & Mohanty, M. (2021). Explainable AI: Current status and future directions. IEEE Access.

[6] Trisal, A., & Mandloi, D. D. (2021). Machine learning: an overview. International Journal of Research - GRANTHAALAYAH, 9(7), 343-348. https://doi.org/10.29121/granthaalayah.v9.i7.2021.4120

[7] Liu, Q., & Wu, Y. (2012). Supervised learning. In Encyclopedia of Biometrics (pp. 1423-1431). Springer US. https://doi.org/10.1007/978-1-4419-1428-6_451

[8] Dridi, S. (2021). Unsupervised Learning - A Systematic Literature Review [Preprint].

[9] Ghasemi, M., & Ebrahimi, D. (2024). Introduction to Reinforcement Learning [Preprint]. arXiv:2408.07712v3 [cs.AI]. https://arxiv.org/abs/2408.07712

[10] Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science, 2(420). https://doi.org/10.1007/s42979-021-00815-1

[11] Islam, M., Chen, G., & Jin, S. (2019). An overview of neural network. American Journal of Neural Networks and Applications, 5(1), 7–11. https://doi.org/10.11648/j.ajnna.20190501.12

[12] Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. E. (2009). Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8(7), 579–588.

[13] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 . https://doi.org/10.48550/arXiv.1511.08458

[14] Schmidt, R. M. (2019). Recurrent neural networks (RNNs): A gentle introduction and overview. arXiv preprint arXiv:1912.05911 . https://doi.org/10.48550/arXiv.1912.05911

[15] Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. arXiv preprint arXiv:1909.09586 . https://doi.org/10.48550/arXiv.1909.09586

[16] Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., & Dahl, G. E. (2020). On empirical comparisons of optimizers for deep learning. arXiv preprint arXiv:1910.05446v3. https://arxiv.org/abs/1910.05446v3

[17] Scott, L. R. (2011). Numerical analysis. Princeton University Press. ISBN: 978-0-691-14686-7

[18] Devi, S., & Jakhar, M. (2018). An introduction to differential equations. International Journal of Statistics and Applied Mathematics, 3(1), 115-117.

[19] Nagy, G. (2021, January 18). Ordinary differential equations . Mathematics Department, Michigan State University. https://users.math.msu.edu/users/gnagy/teaching/ode.pdf

[20] Τσίτσας, Ν. Λ. (2024). Εφαρμοσμένα μαθηματικά: Διαφορικές εξισώσεις, μιγαδικές συναρτήσεις και ανάλυση Fourier. Κλειδάριθμος. ISBN: 978-960-645-676-3.

[21] Hairer, E., & Wanner, G. (2015). Initial Value Problems. In Encyclopedia of Applied and Computational Mathematics (pp. 1-6). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-70529-1_121

[22] Magyar, P. (n.d.). Taylor Series. Math 133, Stewart §11.10. Michigan State University.

[23] Biswas, B. N., Chatterjee, S., Mukherjee, S. P., & Pal, S. (2013). A DISCUSSION ON EULER METHOD: A REVIEW. Electronic Journal of Mathematical Analysis and Applications, 1(2), 294-317.

[24] Butcher, J. C. (1996). A history of Runge-Kutta methods. Applied Numerical Mathematics, 20, 247-260. https://doi.org/10.1016/0168-9274(95)00108-5

[25] Butcher, J. C. (2008). Numerical Methods for Ordinary Differential Equations (2nd ed.). John Wiley & Sons Ltd.

[26] Cartwright, J. H. E., & Piro, O. (1992). The dynamics of Runge-Kutta methods. International Journal of Bifurcation and Chaos, 2, 427-449.

[27] Zhu, M., Chang, B., & Fu, C. (2022). Convolutional Neural Networks combined with Runge–Kutta Methods. arXiv preprint arXiv:1802.08831. https://arxiv.org/abs/1802.08831

[28] Kim, B., Chudomelka, B., Park, J., Kang, J., Hong, Y., & Kim, H. J. (2020). Robust Neural Networks inspired by Strong Stability Preserving Runge-Kutta methods. arXiv preprint arXiv:2010.10047. https://arxiv.org/abs/2010.10047

[29] Wang, T., Li, H., Noori, M., Ghiasi, R., Kuok, S.-C., & Altabey, W. A. (2023). Seismic response prediction of structures based on Runge-Kutta recurrent neural network with prior knowledge. Engineering Structures, 279, 115576. https://doi.org/10.1016/j.engstruct.2022.115576

[30] Zhuang, Q., Lorenzi, J. M., Bungartz, H.-J., & Hartmann, D. (2021). Model order reduction based on Runge–Kutta neural networks. Data-Centric Engineering, 2, e13. https://doi.org/10.1017/dce.2021.15

[31] Zheng, R., Zhang, Y., Huang, D., & Chen, Q. (2020). Sequential Convolution and Runge-Kutta Residual Architecture for Image Compressed Sensing. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, vol 12365 (pp. 229-245). Springer, Cham. https://doi.org/10.1007/978-3-030-58545-7_14