ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

"ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ.ΘΕΩΡΙΑ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΕΛΕΓΧΟΥ "

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΚΑΝΟΝΙΚΩΝ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ. ΕΦΑΡΜΟΓΕΣ ΜΕ ΠΡΟΓΡΑΜΜΑΤΑ FORTRAN. ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΤΩΝ ΜΕΘΟΔΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αγορίτσα Μ. Ρόιδου

Επιβλέπουσα: Μαρία Γουσίδου - Κουτίτα

Αν.Καθηγήτρια Α.Π.Θ.

Θεσσαλονίκη, Ιούνιος 2010

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

" ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ.ΘΕΩΡΙΑ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΕΛΕΓΧΟΥ "

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΚΑΝΟΝΙΚΩΝ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ. ΕΦΑΡΜΟΓΕΣ ΜΕ ΠΡΟΓΡΑΜΜΑΤΑ FORTRAN. ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΤΩΝ ΜΕΘΟΔΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αγορίτσα Μ. Ρόιδου

Επιβλέπουσα: Μαρία Γουσίδου - Κουτίτα

Αν. Καθηγήτρια Α.Π.Θ.

Εγκρίθηκε από την τριμελή
εξεταστική επιτροπή την

……	……	……
Μ. Γουσίδου – Κουτίτα	Α. Βαρδουλάκης	Ν. Καραμπετάκης
Αν.Καθηγήτρια Α.Π.Θ.	Καθηγητής Α.Π.Θ.	Αν. Καθηγητής Α.Π.Θ.

Θεσσαλονίκη, Ιούνιος 2010

……..

Αγορίτσα Μ. Ρόιδου

Πτυχιούχος Μαθηματικός Α.Π.Θ.

ΠΕΡΙΛΗΨΗ

Οι διαφορικές εξισώσεις χρησιμοποιούνται ευρύτατα για την μοντελοποίηση πολλών διαφορικών φυσικών φαινομένων και συστημάτων και κυρίως αυτών που εξελίσσονται στο χρόνο. Η ευρεία χρήση τους ήταν και ο λόγος που μαγνήτισαν την συγγραφέα αυτής της διπλωματικής να ασχοληθεί με αυτές από την σκοπιά της αριθμητικής ανάλυσης. Στην εργασία αυτή έγινε προσπάθεια να παρουσιαστούν όλες οι αριθμητικές μέθοδοι που χρησιμοποιούνται για την επίλυση διαφορικών εξισώσεων πρώτης τάξης με αρχικές τιμές.

Ξεκινώντας από το πρώτο κεφάλαιο, ο αναγνώστης θα συναντήσει μια αναφορά σε εισαγωγικές έννοιες της αριθμητικής ανάλυσης όπως και την απλούστερη αλλά και παλαιότερη μέθοδο, την μέθοδο του Euler.

Στην συνέχεια στο δεύτερο κεφάλαιο μια πληθώρα από μεθόδους παρουσιάζονται σταδιακά. Οι μέθοδοι του Taylor $2^{ης}$ και $4^{ης}$ τάξης που ορίζονται με την βοήθεια των αντίστοιχων πολυωνύμων του Taylor, αμέσως μετά από τις οποίες ανοίγεται ο πλούτος των μεθόδων των οικογενειών Runge- Kutta.Στην οικογένεια των μεθόδων Runge- Kutta $2^{ης}$ τάξης όπου ανήκουν οι μέθοδοι Μέσου Σημείου (Midpoint Method), η Βελτιωμένη Μέθοδος του Euler (Modified Euler) και η μέθοδος του Heun. Παρόλο που η τάξη των μεθόδων είναι ακόμη χαμηλή, οι βελτιώσεις των προσεγγίσεων είναι εμφανής και οι μέθοδοι αυτοί είναι κατάλληλοι για την κατανόηση των μεθόδων μεγαλύτερης τάξης. Μετά την μέθοδο Runge Kutta $3^{ης}$ τάξης, έχουν σειρά οι μέθοδοι $4^{ης}$ όπου εξέχουσα θέση καταλαμβάνει η ομώνυμη μέθοδος που παρέχει και τις καλύτερες προσεγγίσεις από όλες τις υπόλοιπες ,ενώ αξιόλογες είναι και οι μέθοδοι Kutta και Gill. Περιορισμένη αναφορά γίνεται στις μεθόδους ανώτερης τάξης όπως και στις μεθόδους μεταβλητού βήματος, που αν και είναι ιδιαίτερα πολύπλοκες αξίζει τον κόπο να καταπιαστούμε μαζί τους κυρίως για την μεγάλη ακρίβεια που παρέχουν.

Το τρίτο και τελευταίο κεφάλαιο, παρουσιάζονται οι πολυβηματικές και οι μέθοδοι πρόβλεψης-διόρθωσης. Η κορυφή των αριθμητικών μεθόδων για την επίλυση διαφορικών εξισώσεων που χρησιμοποιούν τις ήδη υπολογισμένες προσεγγιστικές τιμές, τις οποίες διορθώνουν ελαττώνοντας το σφάλμα αποκοπής.

Κατά την διάρκεια αυτής της διαδρομής θα συναντήσει κανείς ,για κάθε μια μέθοδο, παραδείγματα , σύντομη θεωρία, και αλγορίθμους για την μεταφορά των μεθόδων αυτών σε προγραμματιστικό περιβάλλον , όπως και εφαρμογές τους σε πεδία άλλων επιστημών (π.χ. φυσικής, χημείας, κοινωνιολογίας, κ.α.)

Η ελάχιστη συνεισφορά της συγγραφέας έναντι στον πλούτο των γνώσεων που της προσέφερε η επαφή της με το κομμάτι αυτό της αριθμητικής ανάλυσης βρίσκεται στο παράρτημα. Παραθέτονται προγράμματα όλων των προαναφερθέντων μεθόδων σε προγραμματιστικό περιβάλλον Fortran.6. Η γλώσσα αυτή επιλέχτηκε κυρίως για την ακρίβειά της.

ΛΕΞΕΙΣ- ΚΛΕΙΔΙΑ

Αριθμητική Ανάλυση, Διαφορικές Εξισώσεις, Μέθοδος Euler, Μέθοδος Taylor $2^{ης,}$, Μέθοδος Taylor $4^{ης,}$, Runge- Kutta $2^{ης}$, Modified Euler, Midpoint Method, Heun Method, Runge- Kutta $4^{ης}$,Μέθοδος Kutta, Μέθοδος Gill, Μέθοδος Fehlberg $5^{ης}$, Πολυβηματικές Μέθοδοι.

ABSTRACT

The Differential equations are widely used for many different natural phenomena and systems, and especially those that evolve over time. The author ‘s magnetization by the widespread use was the reason of this diplomatic deal with them ,from the perspective of numerical analysis. In this paper attempted to present all the numerical methods for solving differential equations first order with initial values.

At the first chapter, the reader encounters a reference to introductory concepts of numerical analysis aw the simplest and oldest method, Euler’s method.

In the second chapter a variety of methods are presented gradually. The methods of Taylor 2nd and 4th order defined with the help of the respective polynomials Taylor, immediately after which the presents the wealth of Runge- Kutta methods . The class of Runge-Kutta second order methods contains Midpoint Method, the improved method of Euler (Modified Euler) and the method of Heun. Although the order of methods is still low, these approaches are obvious improved and these methods are suitable for understanding higher-order techniques. After the Runge-Kutta method of third order, fourth order Runge Kutta method occupies a prominent position among the rest methods the same order and that method provides the best the best approaches from all the others, method of Kutta and Gill method are both remarkable. Little mention is made in the methods of higher order as in the methods of variable pitch, which although are very complex worth trouble grips with them mainly to the high accuracy they provide.

The third and final chapter presents the multisteps methods and the predictor-corrector methods. These methods are at the top of numerical methods for solving differential equations using. They use the computed approximations , and correct them.

During this journey you will meet, for each method, examples, brief theory and algorithms for the transportation of these methods in programming environment, as well as applications in other fields of science (eg physics, chemistry, sociology etc.)

The minimum contribution from writer’s side towards the wealth of knowledge offered by the contract with this part of numerical analysis is at annex. We present programs of all the above methods in programming environment Fortran.6. This programming language was chosen primarily for its accuracy.

KEY WORDS

Numerical Analysis, Differential Equations, Method of Euler, Taylor 2nd order method,Taylor fourth order method, Runge-Kutta 2nd, Modified Euler, Midpoint Method, Heun Method, Runge-Kutta fourth order method, Method Kutta, Method Gill, Method Fehlberg 5th, multistep Methods

Πρόλογος

Προλογίζοντας την διπλωματική μου διατριβή θα ήθελα να εκφράσω τον σεβασμό και τις θερμές ευχαριστίες μου προς τους διδάσκοντές μου, που μου έδωσαν την ευκαιρία να φοιτήσω στο συγκεκριμένο μεταπτυχιακό πρόγραμμα σπουδών και να έρθω σε επαφή με το μεγαλείο αυτού του κλάδου. Επίσης οφείλω ένα μεγάλο ευχαριστώ στους συμφοιτητές και συναδέλφους που βαδίσαμε όλοι μαζί αυτά τα δύο χρόνια τα μονοπάτια της γνώσης.

Δράττοντας την ευκαιρία θα ήθελα να απευθύνω θερμές ευχαριστίες στην κα Μαρία Γουσίδου Κουτίτα που τόσο από το έδρανο της καθηγήτριας, όσο και από την θέση της επιβλέπουσας παρείχε γνώσεις και πάντοτε ένα πλατύ χαμόγελο. Οι υποδείξεις της, το επιστημονικό υλικό που μου προσέφερε, το διαρκές ενδιαφέρον της, η αμέριστη συμπαράσταση, οι ώρες που μου αφιέρωσε καθώς και η υπομονή της στην αδιόρθωτη κωλυσιεργία μου έπαιξαν καίριο ρόλο στην εκπόνηση της εργασίας. Επίσης θα ήθελα να ευχαριστήσω τον κ. Καραμπετάκη Νικόλαο για τις άμεσες και κατατοπιστικές συμβουλές του στα προγραμματιστικά προβλήματα που αντιμετώπισα καθ' όλη την διάρκεια την εργασίας.

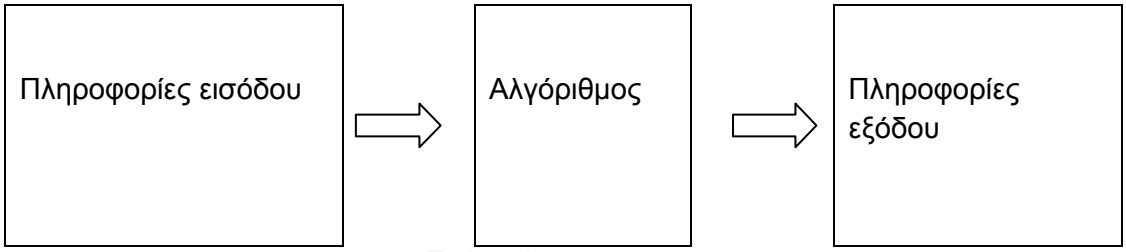
Θερμές ευχαριστίες οφείλω στα μέλη της τριμελούς επιτροπής, τον καθηγητή κ. Βαρδουλάκη Αντώνιο – Ιωάννη, και τον αναπληρωτή καθηγητή κ. Καραμπετάκη Νικόλαο για τον χρόνο που αφιέρωσαν στη μελέτη καθώς και την αξιολόγηση της εργασίας.

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 1^{ο} ΕΙΣΑΓΩΓΙΚΑ

1.0 Τι είναι η αριθμητική ανάλυση.

Η αριθμητική ανάλυση είναι ο τομέας των μαθηματικών που περιλαμβάνει την ανάλυση και την αξιολόγηση μεθόδων για τον υπολογισμό αριθμητικών αποτελεσμάτων από αριθμητικά δεδομένα. Επομένως η αριθμητική ανάλυση είναι ένα είδος επεξεργασίας πληροφοριών. Τα δεδομένα αποτελούν τις πληροφορίες εισόδου, τα αποτελέσματα τις πληροφορίες εξόδου και η μέθοδος υπολογισμού του αλγόριθμου. Αυτά είναι τα βασικά συστατικά ενός προβλήματος αριθμητικής ανάλυσης όπως δίνονται στα διαγράμματά του σχ. 1.



Σχήμα 1

Το κεφάλαιο χωρίζεται σε δύο μέρη. Στο πρώτο μέρος δίνονται ορισμένα στοιχεία από την θεωρία των προβλημάτων αρχικών τιμών για συνήθεις διαφορικές εξισώσεις. Στο δεύτερο και μεγαλύτερο μέρος μελετάται αρκετά λεπτομερώς η μέθοδος Euler. Η μέθοδος αυτή αν και είναι απλή, δεν είναι χρήσιμη στην πράξη.

Ο σκοπός μας είναι η εξοικείωση με τις μεθόδους αριθμητικής ανάλυσης για την λύση διαφορικών εξισώσεων.

1.1 Βασικές έννοιες και ορισμοί.

Παραθέτουμε βασικά στοιχεία από την θεωρία κανονικών διαφορικών εξισώσεων που συνδέονται και συνοδεύουν τις αριθμητικές μεθόδους.

Ορισμός 1.1

Μια συνάρτηση f(x,y) ικανοποιεί μια συνθήκη Lipschitz ως προς τη μεταβλητή y πάνω στο σύνολο $D \subset \mathbb{R}^2$ αν υπάρχει μια σταθερός $L > 0$ που ικανοποιεί τη σχέση

$$\left| f(x,y_1) - f(x-y_2) \right| \leq L \left| y_1 - y_2 \right| \qquad (1.1)$$

Για κάθε (x_1y_1) και $(x_1y_2) \in D$

Η σταθερά L ονομάζεται σταθερά Lipschitz για την f

Παράδειγμα 1.1

Εάν D = {(t,y) : 1 ≤ t ≤ 2 , -3 ≤ y ≤ 4 } και f(t,y) = t│y│ τότε για κάθε ζεύγος (t,y_1) , (t,y_1) ∈ D ισχύει

$$\left|f(t,y_1) - f(t,y_2)\right| = \left| t\left| y_1\right| - t_1\left|y_2\right| \right| = \left|t\right|\left|\left| y_1\right|-\left|y_2\right|\right| \le 2 \left|y_1-y_2\right| \qquad (1.2)$$

Επομένως, η συνάρτηση f ικανοποιεί την συνθήκη Lipschitz στο D για την μεταβλητή y με

L=2

■

Ορισμός 1.2

Ένα σύνολο $D \subset \mathbb{R}^2$ ονομάζεται κυρτό αν για κάθε (x_1y_1) , $(x_2y_2) \in D$ το σημείο $((1-\lambda) + \lambda \chi_2, (1-\lambda)y_2)$ ανήκει επίσης στο D για κάθε λ , με 0≤ λ ≤ 1

Από γεωμετρική άποψη ο ορισμός 2 ορίζει ότι ένα σύνολο D είναι κυρτό σημαίνει ότι για οποιαδήποτε δύο σημεία, που ανήκουν στο σύνολο τότε το ευθύγραμμο τμήμα με άκρα τα σημεία αυτά πρέπει επίσης να ανήκει στο σύνολο D. Τα σύνολα που θα θεωρήσουμε είναι της μορφής D={(t,y) : a ≤ t ≤ b , -∞ < y <∞} όπου a, b σταθερές.

Άσκηση 1.1

Να αποδειχθεί ότι για κάθε σταθερά a, b το σύνολο

D={(t,y) / a ≤ t ≤ b , -∞<y< ∞} είναι κυρτό.

Λύση:

Αρκεί να αποδειχθεί ότι το σημείο $A((1-\lambda)t_1 + \lambda_2 , (1-\lambda)y, + \lambda y_2) \in D$ όταν τα σημεία $K(t_1, y_1)$ και $\Lambda(t_2,y_2) \in D$.

$0 \le \lambda \le 1 \Rightarrow 1 - \lambda > 0$

$K \in D \Rightarrow \alpha \le t_1 \le b \Rightarrow (1-\lambda)\, a \le (1-\lambda)t_1 \le (1-\lambda)b$ (1)

$\Lambda \in D \Rightarrow \alpha \le t_2 \le b \Rightarrow \lambda a \le \lambda+2 \le \lambda b$ (2)

Προσθέτοντας κατά μέλη τις ανισότητες (1) και (2) προκύπτει:

$(1-\lambda)\, a + \lambda\alpha \le (1-\lambda)\, t_1 + \lambda t_2 \le b$ (3)

Ομοίως, $K \in D \Rightarrow -\infty < y_1 < \infty \Rightarrow -\infty < (1-\lambda)y < \infty$ (4)

$\Lambda \in D \Rightarrow \infty < y_2 < \infty \Rightarrow -\infty < \lambda y_2 < \infty$ (5)

Προσθέτοντας κατά μέλη τις ανισότητες (4) και (5) προκύπτει :

$-\infty < (1-\lambda)y_1 + \lambda y_2 < \infty$ (6)

Από τις σχέσεις (3) και (6) προκύπτει ότι

$((1-\lambda)t_1 + \lambda t_2, (1-\lambda)y_1 + \lambda y_2) \in D$

Δηλαδή $A \in D$.

■

Θεώρημα 1.1

Έστω συνάρτηση f(x, y) ορισμένη σε ένα κυρτό σύνολο $D \subset \mathbb{R}^2$.

Εάν υπάρχει μια θετική σταθερά L και $\left| \frac{df}{dx}(t,y) \right| \leq L$

για κάθε $(t, y) \in D$

τότε η συνάρτηση f ικανοποιεί μια συνθήκη Lipschitz L.

■

Θεώρημα 1.2

Θεωρούμε το σύνολο $D=\{(t, y) : a \leq t \leq b, -\infty < y < \infty\}$

Και η συνάρτηση f(t,y) είναι συνεχής στο D. Εάν η f ικανοποιεί μια συνθήκη Lipschitz στο D για την μεταβλητή y, τότε το πρόβλημα αρχικών τιμών

$y'=f(t,y)$, $a \leq t \leq b$ και $y(a)=a$ (1.3)

έχει μοναδική λύση y(t) για $a \leq t \leq b$.

■

Ορισμός 1.3

Το πρόβλημα αρχικών τιμών

. $y'=f(t,y)$, $a \leq t \leq b$ και $y(a)=a$

ονομάζεται καλά τοποθετημένο εάν:

- Υπάρχει μία μοναδική λύση για το πρόβλημα
- Υπάρχει αριθμός $\varepsilon>0$ με την ιδιότητα ότι υπάρχει μία μοναδική λύση u(x) του προβλήματος

$u'=f(t,y)+\delta(\chi)$, $a \le x \le b$ και $u(a)=y_0+\varepsilon_0$ (1.4)

- για κάθε $|\varepsilon_0|<\varepsilon$ και $|\delta(\chi)|<\varepsilon$ και για κάθε $\chi \in [a,b]$
- Υπάρχει κ>0 έτσι ώστε

$|u(x)-y(x)|<k\varepsilon$ για κάθε $x \in [a,b]$ (1.5)

■

Ορισμός 1.4

Μια εξίσωση της μορφής

$f(x,y,\frac{\mathbf{dy}}{\mathbf{dx}},\frac{d^2y}{dx^2}\ldots,\frac{d^ny}{dx^n})$

ονομάζεται κανονική διαφορική εξίσωση n τάξης, όταν περιέχει ολικά διαφορικά n-τάξης και έχει μία μόνο ανεξάρτητη μεταβλητή που συνήθως είναι η x, ενώ η y είναι η εξαρτημένη μεταβλητή.

Μία συνάρτηση y(x), που ικανοποιεί την εξίσωση, ονομάζεται λύση της διαφορικής όταν την επαληθεύει και υπολογίζονται οι παράγωγοι τουλάχιστον n- τάξεως.

■

Για μερικές διαφορικές εξισώσεις υπάρχουν αναλυτικές λύσεις, ενώ στις περισσότερες η αναλυτική λύση είναι πολύ περίπλοκη π.χ. είναι άθροισμα με άπειρους όρους ή δεν μπορεί να βρεθεί. Στις περιπτώσεις αυτές χρησιμοποιούμε τις αριθμητικές μεθόδους. Για να πετύχουμε μια μοναδική αριθμητική λύση πρέπει να δοθούν, εκτός από την διαφορική εξίσωση και άλλες πληροφορίες όπως οι τιμές της y(x) για ορισμένες τιμές της x. Για την διαφορική εξίσωση n- τάξης αυτές οι συνθήκες είναι αρκετές, για να προσδιορίσουν μία μοναδική λύση.

Αν όλες οι συνθήκες προσδιορίζονται για την ίδια τιμή του x έχουμε ένα πρόβλημα αρχικών τιμών (initial value problem). Αν οι συνθήκες προσδιορίζονται για περισσότερες τιμές του x (π.χ. για την x_0 και την x_n), έχουμε πρόβλημα οριακών τιμών (boundary value problem).

Στα πλαίσια αυτής της διπλωματικής εργασίας θα μελετήσουμε και θα αναπτύξουμε μερικές από τις πιο κλασσικές μεθόδους της Αριθμητικής Ανάλυσης που δίνουν τον υπολογισμό μιας προσεγγιστικής τιμής της άγνωστης συνάρτησης y σε ένα τυχαίο σημείο $x \neq x_0$. Οι μέθοδοι που θα περιγραφούν χαρακτηρίζονται ως άμεσοι καθώς χρησιμοποιούν κατευθείαν την διαφορική εξίσωση και την αρχική συνθήκη της και όχι τυχόν υπάρχουσα αναλυτική έκφραση για την λύση.

Για την αριθμητική επίλυση του παραπάνω προβλήματος εργαζόμαστε ως εξής. Επιλέγουμε αυθαίρετα έναν φυσικό αριθμό n και συμβολίζουμε με x_n την τιμή του x για την οποία ζητείται η τιμή της y. Στη συνέχεια διαιρούμε το διάστημα $[x_0, x_n]$ (αν υποθέσουμε ότι $x_n>x_0$, διαφορετικά δουλεύουμε αναλόγως) σε n ίσα υποδιαστήματα με τα σημεία x_i, όπου i=0,n μπορούν αν οριστούν από τις σχέσεις x_n-x_0

$$x_i=x_0+ih \quad i=0,n \ , \ h= (x_n-x_0)/n$$

με h το βήμα ολοκλήρωσης. Στην συνέχεια εφαρμόζοντας μια από τις αριθμητικές μεθόδους, που θα αναπτύξουμε παρακάτω, βρίσκουμε την τιμή y_i που είναι μια προσεγγιστική τιμή της αληθούς λύσης. Μετά χρησιμοποιώντας την y_1 και όχι την $y(x_1)$ με την ίδια μέθοδο την y_2 που είναι μια προσεγγιστική τιμή της αληθούς τιμής $y(x_2)$ κ. ο. κ και τέλος την y_n, την οποία δεχόμαστε ως την αριθμητική λύση του προβλήματος στο σημείο $x=x_n$. Στην πράξη ο φυσικός αριθμός n δεν επιλέγεται τελείως αυθαίρετα. Συνήθως παίρνουμε n=1 και βρίσκουμε την τιμή y_n $(=y_1)$ εφαρμόζοντας την αριθμητική μέθοδο, έπειτα παίρνουμε n=2 και βρίσκουμε, με την ίδια μέθοδο, την $y_n(=y_2)$. Εργαζόμενοι με τον τρόπο που περιγράψαμε προηγούμενα συνεχίζουμε για n=3, n=4, κ.ο.κ. Σταματούμε την διαδικασία, όταν βρούμε δύο διαδοχικές τιμές για την y_n, που να συμφωνούν σε ένα προκαθορισμένο πλήθος δεκαδικών ή σημαντικών ψηφίων. Ως αριθμητική λύση της διαφορικής εξίσωσης δεχόμαστε την τελευταία τιμή y_n στο σημείο $x=x_n$.

Οι μέθοδοι που θα αναπτυχθούν στη συνέχεια διακρίνονται σε δύο μεγάλες κατηγορίες. Σε αυτές, που για την εύρεση της τιμής y_{i+1} για i=1,…,n-1 χρησιμοποιείται μία εξίσωση διαφορών πρώτης τάξης, λυμένη ως προς y_{i+1} , που γενικά δεν είναι ούτε γραμμική ούτε έχει σταθερούς συντελεστές και ονομάζονται μέθοδοι απλού βήματος. Επίσης σε αυτές που για την εύρεση της y_{i+1} για i=1,…,n-1 χρησιμοποιείται μια εξίσωση διαφορών μεγαλύτερης τάξης λυμένη ως προς y_{i+1} , και ονομάζονται αντίστοιχα μέθοδοι πολλαπλού βήματος. Τέλος θα πρέπει να τονιστεί ότι στις περισσότερες μεθόδους γίνεται χρήση του αναπτύγματος του Taylor.

1.2 Η μέθοδος Euler

Η μέθοδος Euler που θα παρουσιαστεί στην συνέχεια, χρησιμοποιείται σπάνια, παρόλο που είναι μια πάρα πολύ απλή μέθοδος. Η απλότητα της παραγωγής της μεθόδου αυτής μπορεί να χρησιμοποιηθεί για να τονίσει τις τεχνικές που υπάρχουν στην κατασκευή μερικών πιο προηγμένων τεχνικών, χωρίς την χρήση δυσκίνητης άλγεβρας, που ακολουθεί αυτές τις κατασκευές.

Η μέθοδος πραγματεύεται την επίτευξη μιας προσέγγισης στο καλά τοποθετημένο πρόβλημα αρχικών τιμών

$$. \quad y'=f(t,y) \quad , a \le t \le b \text{ και } y(a)=a$$

Στην πραγματικότητα, μια συνεχή προσέγγιση της λύσης x(t) δεν μπορεί επιτευχθεί. Αντί αυτού προσεγγίσεις του y μπορούν να βρεθούν σε πολλά σημεία του διαστήματος [a,b], τα οποία ονομάζονται mesh points. Καθώς η προσέγγιση της λύσης σε αυτά τα σημεία επιτευχθεί, η προσεγγιστική λύση στα υπόλοιπα σημεία του διαστήματος [a,b] μπορεί να υπολογιστεί χρησιμοποιώντας κάποια από τις μεθόδους παρεμβολής(π.χ. παρεμβολή Lagrange, Newton, Hermite)

Τα κομβικά σημεία (mesh points**)** γίνεται η παραδοχή να μην είναι τυχαία σημεία του διαστήματος. Τα σημεία αυτά είναι ομοιόμορφα κατανεμημένα μέσα στο διάστημα [a,b] και ισαπέχουν μεταξύ τους. Επιλέγεται ένας φυσικός αριθμός που συμβολίζεται με N και επιλέγουμε τα σημεία $\{t_0 , t_1 , t_2, \ldots, t_N \}$ για τα οποία ισχύει

$$t_i = a + ih \quad \text{όπου } i=0,1,2,\ldots,N$$

Η απόσταση μεταξύ των σημείων h=(b-a)/N ονομάζεται βήμα (step size) .

Υποθέτουμε ότι η y(t) έχει δύο συνεχής παραγώγους στο [a,b] τέτοιε ώστε για κάθε i=0,1,2…,N η $y (t_{i+1})$ να γράφεται ως εξής:

$$y(t_{i+1}) = y(t_i + h) = y(t_i) + h\, y'(t_i) + \frac{h^2}{2} y''(\xi_i)$$

για κάποια σημεία ξ_i, τέτοια ώστε $t_i < \xi_i < t_{i+1}$.

Χρησιμοποιώντας ότι $h = t_{i+1} – t_i$ προκύπτει ότι ένας αριθμός θ_i υπάρχει όπου

$0 < \theta_i < 1$

και μάλιστα ισχύει ότι

$$y(t_{i+1}) = y(t_i) + h\, y'(t_i) + h^2/2\, y''(t_i + \theta_i h)$$

και καθώς η y(t) ικανοποιεί την διαφορική εξίσωση προκύπτει

$$y(t_{i+1}) = y(t_i + h) = y(t_i) + h\, f(t_i , y(t_i)) + h^2/2\, y''(t_i + \theta_i h).$$

Ξαναγράφοντας την ισότητα:

$$\frac{y (t_{i+1}) - y(t_i)}{h} = f(t_i , y(t_i)) + \frac{h}{2} y''(t_i + \theta_i h) \text{ για } i=0,1,2,N$$

Όταν το βήμα h γίνει αρκετά μικρό η συνέχεια της y'' συνεπάγεται ότι ο όρος

$\frac{h}{2} y''(t_i + \theta_i h)$ είναι επίσης μια μικρή ποσότητα και για αυτό τον λόγο

$\frac{y(t_{i+1}) - y(t_i)}{h} \approx f(t_i, y(t_i))$

Η μέθοδος του Euler χρησιμοποιεί αυτή την υπόθεση και προσδιορίζει

$w_0 = a$ (1.6)

$w_{i+1} = w_i + h\, f(t_i, w_i)$

υποθέτοντας ότι $w_i \approx y(t_i)$ για κάθε i=0,1,2,…,N

Η εξίσωση (1.5) ονομάζεται εξίσωση διαφορών της μεθόδου Euler .

Αλγόριθμος του Euler

- Εισαγωγή των άκρων του διαστήματος [a,b] και του πλήθους των διαστημάτων N
- Προσδιορισμός της αρχικής συνθήκης
- w=a
- Υπολογισμός του βήματος h από την σχέση
- h=(b-a)/N
- Προσδιορισμός του πρώτου σημείου t = a,δηλ με το αριστερό άκρο του διαστήματος
- Καθώς (Do) i=0,1,2,…,N υπολογισμός των
- w = w + hf(t,w) (καινούριο w)
- t=t+h (καινούριο t)
- Εκτύπωση των (t, w).
- Τέλος

Παράδειγμα 1.2

Να βρεθούν οι προσεγγίσεις του προβλήματος αρχικών τιμών για την συνάρτηση

$y' = -y + t + 1, \quad 0 \le t_i \le 1, \quad y(0) = 1$

όπου N=10, επομένως $h = (b - a) / N = (1 - 0)/ 10 = 0.1$

Λύση:

Χρησιμοποιούμε την συνάρτηση f(t , y) = - y + t + 1

Η αρχική συνθήκη είναι w_0=1 και i = 1 ,…, 10.

Εύκολα αποδεικνύεται ότι η ακριβής λύση είναι η συνάρτηση y(t) = t + e^{-t}.

Κομβικά σημεία	**Προσεγγιστικές λύσεις**	**Ακριβείς λύσεις**	**Σφάλμα Euler**
0	1	1	0,00
0.1	1	1,004837	0,004837
0.2	1,100000	1,018731	0,008731
0.3	1,029000	1,040818	0,011818
0.4	1,056100	1,070320	0,014220
0.5	1,090490	1,106531	0,016041
0.6	1,131441	1,148812	0,017371
0.7	1,178297	1,196585	0,018288
0.8	1,230467	1,249329	0,018862
0.9	1,287420	1,306570	0,019150
1	1,348678	1,367879	0,019201

Στον παραπάνω πίνακα παραθέτονται οι προσεγγιστικές τιμές w_i στα σημεία ti , οι ακριβής λύσεις στα ίδια σημεία και το σφάλμα. Οι προσεγγιστικές λύσεις είναι τα αποτελέσματα του προγράμματος EulerMethod που έχει κάνει η συγγραφέας της διπλωματικής αυτής σε Fortran. Το πρόγραμμα υπάρχει στο παράρτημα. ■

Σημείωση:

Το σφάλμα αυξάνεται ελαφρώς καθώς οι τιμές των t_i αυξάνονται. Αυτή η αύξηση είναι αποτέλεσμα της σταθερότητας της μεθόδου Euler, το οποίο συνεπάγεται ότι τα σφάλματα στρογγυλοποιήσεων αναμένεται να αυξάνονται, όχι περισσότερο από γραμμικά.

Με σκοπό να βρούμε ένα όριο για τα σφάλματα της γενικής περίπτωσης της μεθόδου Euler, είναι απαραίτητο να παρουσιαστούν τα παρακάτω λήμματα.

Λήμμα 1.1:

Για κάθε $x \geq -1$και για κάθε θετικό ακέραιο n, ισχύει:

$$0 \leq (1 + x)^n \leq e^{nx}$$

Λήμμα 1.2:

Εάν m και n είναι δύο πραγματικοί θετικοί και $\{a_i\}_{i=0}^k$ είναι μία ακολουθία τέτοια ώστε $a_0 \geq -n/m$ και $a_{i+1} \leq (1 + m) a_i + n$ για κάθε i = 0, 1, 2, …, k τότε

$$\mathbf{a_{i+1}} \leq \mathbf{e}^{(i+1)m} \left(\frac{n}{m} + \mathbf{a_0} \right) - \frac{n}{m}.$$

Θεώρημα 1.3:

Έστω y (t) η μοναδική λύση στο καλά τοποθετημένο πρόβλημα αρχικών τιμών

. $y'=f(t ,y)$, $a \leq t \leq b$ και $y(a)=a$

και $w_0, w_1, w_2, \ldots, w_N$ οι προσεγγίσεις της μεθόδου Euler για κάποιον θετικό ακέραιο N. Εάν η f ικανοποιεί μία συνθήκη Lipschitz με σταθερά L στο

$D = \{ (t, y) : a \leq t \leq b , -\infty < y < +\infty \}$

και υπάρχει μία σταθερά M με την ιδιότητα

$| y'' (t) | \leq M$ για κάθε t ανήκει [a, b] τότε

$| y (t_i) – w_i | \leq \frac{hM}{2L} [e^{L(t_i-a)} – 1]$ για κάθε i = 0, 1, 2, …, N.

1.3 Παραδείγματα – εφαρμογές

Εφαρμογή 1.1:

Σε ηλεκτρικό κύκλωμα με ένταση $\mathcal{E}$, αντίσταση R , επαγωγή L , χωρητικότητας C σε παραλληλία, όπου η μεταβλητή i ικανοποιεί την διαφορική εξίσωση

$$\frac{di}{dt} = C \frac{d^2 \mathcal{E}}{dt^2} + \frac{1}{R} \frac{d\mathcal{E}}{dt} + \frac{1}{L} \mathcal{E}$$

Εάν C=0.3farads , R1.4 ohms , L=1.7henries και η ένταση δίνεται από τον τύπο

$$\varepsilon(t)=e^{-0.06\pi t}\sin(2t-\pi).$$

Εάν i(0)=0, να βρεθεί το τρέχον i για τις τιμές t=0.1j για j=0, 1, …, 100 χρησιμοποιώντας την μέθοδο του Euler.

Λύση:

Στον πίνακα που ακολουθεί παρουσιάζονται περιληπτικά οι προσεγγιστικές λύσεις του παραπάνω προβλήματος που υπολογίστηκαν με το πρόγραμμα EulerMethodForVoltage σε Fortran που δημιούργησε η συγγραφέας αυτής της διπλωματική. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα.

κομβικά σημεία	προσεγγιστική λύση
0	1
1	0,1219599
2	-2,512025
3	-1,287270
4	3,867631
5	-3,648375
6	-4,285908
7	7,209725
8	-2,859874
9	-8,475985
10	9,489725

■

Εφαρμογή 1.2:

Στο βιβλίο με τίτλο " Looking at History Through Mathematics," Rashevsky (pages 103-110) θεωρείται ένα μοντέλο που περιγράφει την παραγωγή αντικομφορμιστών στην κοινωνία. Υποθέτουμε ότι η κοινωνία έχει πληθυσμό x(t) ατόμων την χρονική στιγμή t, σε χρόνια, και όλοι οι αντικομφορμιστές που έρχονται σε επαφή με άλλους αντικομφορμιστές γεννούν αντικομφορμιστές, ενώ ένα σταθερό ποσοστό r από τις υπόλοιπες γέννες είναι επίσης αντικομφορμιστές. Εάν οι δείκτες γεννήσεων και θανάτων για όλα τα άτομα θεωρούνται ότι είναι οι σταθερές b και d αντίστοιχα.

Υποθέτοντας ότι οι κομφορμιστές και οι αντικομφορμιστές γεννιούνται τυχαία, τότε το πρόβλημα μπορεί να εκφραστεί από τις διαφορικές εξισώσεις

$$\frac{dx(t)}{dt} = (b-d)x(t) \quad \text{και} \quad \frac{dxn(t)}{dt}=(b-d)x_n(t)+rb(x(t)-x_n(t)),$$

Όπου $x_n(t)$ δείχνει το πλήθος των αντικομφορμιστών στον πληθυσμό την χρονική στιγμή t . Η μεταβλητή $p(t)= \frac{xn(t)}{x(t)}$ αντιπροσωπεύει το ποσοστό των αντικομφορμιστών στην κοινωνία την χρονική στιγμή t, η οποία όταν χρησιμοποιηθεί τότε οι παραπάνω εξισώσεις εάν συνδυαστούν απλοποιούνται στην επόμενη διαφορική εξίσωση

$$\frac{dp(t)}{dt} = rb(1-p(t))$$

Εάν p(0)-0.01 , b=0.02, b=0.02 , d=0.015 και r=0.1 να επιλυθεί το παραπάνω πρόβλημα στο διάστημα [0 , 50] με βήμα h=1 χρόνος.

Λύση:

Στον πίνακα που ακολουθεί εμφανίζονται περιληπτικά οι προσεγγιστικές λύσεις του προβλήματος όπως αυτές προέκυψαν από το πρόγραμμα που δημιούργησε η συγγραφέας αυτής της διπλωματικής σε Fortran. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα.

κομβικά σημεία ti	**Προσεγγιστική λύση**
0	0,01
1	$1,2*10^{-2}$
5	$-3,6880374*10^{-10}$
10	$-5,9999999*10^{-2}$
15	-0,17
20	-0,33
25	-0,54
30	-0,8
35	-1,11
40	-1,47
45	-1,88
50	-2,34

ΚΕΦΑΛΑΙΟ 2 ο ΜΕΘΟΔΟΙ ΑΝΩΤΕΡΗΣ ΤΑΞΗΣ

2.1 Εισαγωγή

Καθώς το αντικείμενο των αριθμητικών μεθόδων είναι γενικά, ο προσδιορισμός καλών προσεγγίσεων με την μικρότερη προσπάθεια. Είναι αναγκαίο να έχουμε ένα μέτρο σύγκρισης για την αποδοτικότητα των διαφόρων προσεγγιστικών μεθόδων σεβόμενοι τη χρήση τους στους υπολογισμούς. Στις τεχνικές επίλυσης κανονικών διαφορικών εξισώσεων, όπως η μέθοδος Euler, εισάγεται το πρώτο μέτρο σύγκρισης το τοπικό σφάλμα αποκοπής. Το τοπικό σφάλμα αποκοπής μετράει σε κάθε βήμα της προόδου το μέγεθος της απόκλισης της προσεγγιστικής τιμής από την ακριβή τιμή.

Στη μέθοδο του Euler σε κάθε βήμα του προβλήματος το τοπικό σφάλμα αποκοπής είναι:

$\tau_{(i)} = \frac{yi-yi+1}{h}$ - $f(t_{i-1}, y_{i-1})$ για κάθε i = 1,2,3,…,N

Όπου $y_i = y(t_i)$ είναι η ακριβής τιμή της λύσης στο σημείο t_i. Το σφάλμα αυτό ονομάζεται τοπικό σφάλμα καθώς μετράει την ακρίβεια της μεθόδου σε κάθε βήμα, υποθέτοντας ότι η μέθοδος ήταν ακριβής στο προηγούμενο βήμα. Ως εκ τούτου, το σφάλμα εξαρτάται από τη διαφορική εξίσωση, το βήμα h, και τις προσεγγιστικές τιμές.

Μάλιστα αποδεικνύεται ότι :

$\tau_i = \frac{h}{2} y''(t_i+\theta_i h)$ για κάποιο θ_i τέτοιο ώστε $0 < \theta_i < 1$

Όταν υπάρχει και είναι γνωστή η δεύτερη παράγωγος της συνάρτησης και είναι φραγμένη από μία σταθερά M στο διάστημα [a,b] τότε προκύπτει ότι:

$|\tau_i| \leq \frac{h}{2} M$

Δηλαδή το φράγμα αποκοπής είναι και αυτό με τη σειρά του φραγμένο. Συμπεραίνουμε ότι το σφάλμα αποκοπής της μεθόδου Euler είναι O(h).

2.2 Η Μέθοδος του Taylor

Η μέθοδος του Euler που παρουσιάστηκε στο προηγούμενο κεφάλαιο προήλθε χρησιμοποιώντας το θεώρημα Taylor για n=2. Επεκτείνοντας αυτήν την τεχνική παραγωγής σε μεγαλύτερες τιμές του n , προκύπτουν μέθοδοι που βελτιώνουν την σύγκλιση.

Θεωρούμε ότι η λύση y(t) στο πρόβλημα αρχικών τιμών

$y' = f(t,y)$, $a \le t \le b$, $y(\alpha)=\alpha$

έχει (n+1) συνεχείς παραγώγους και επεκτείνοντας τη λύση y(t) από την άποψη ενός n-οστού βαθμού πολυωνύμου Taylor γύρω από τα κομβικά σημεία t_i , λαμβάνουμε:

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \ldots + \frac{h^n}{n!}y^{(n)}(t_i) + + \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(t_i+\theta_i h) \quad (2.1)$$

για κάποιο θ_i , $0 \le \theta_i \le 1$

Διαδοχικές παραγωγήσεις της λύσης y(t) δίνουν :

$y'(t) = f(t,y(t))$

$y''(t) = f'(t,y(t))$

και γενικά:

$y^{(k)}(t) = f^{(k-1)}(t,y(t))$

Αντικαθιστώντας τα στην (2.1) προκύπτει:

$$y(t_{i+1}) = y(t_i) + h\, f(t_i, y(t_i)) + \frac{h^2}{2} f'(t_i, y(t_i)) + \ldots$$

$$+ \frac{h^n}{n!} f^{(n+1)}(t_i, t_i)) + + \frac{h^{n+1}}{(n+1)!} f^{(n)}(t_i+\theta_i h, g(t_i+\theta_i h)) \quad (2.2)$$

για κάποιο θ_i τ.ω $0 \le \theta_i \le 1$

Η διαφορική μέθοδος που περιγράφεται από την σχέση (2.2) λαμβάνεται παραμελώντας τον όρο που περιλαμβάνει το θ_i. Αυτή η μέθοδος ονομάζεται μέθοδος του Taylor τάξης n.

$w_0=\alpha$

$w_{(i+1)} = w_i + hT^{(n)}(t_i , w_i)$ i=0,1,…,N-1 (2.3)

όπου $T^{(n)}(t_i , w_i) = f(t_i , w_i) + \frac{h}{2}f'(t_i,w_i) + \ldots + \frac{h^{n-1}}{n!} f^{(n-1)}(t_i,w_i)$ (2.4)

Σημειώνεται ότι με αυτήν την ορολογία, η μέθοδος του Euler είναι μέθοδος του Taylor πρώτης τάξης (n=1).

Μέθοδος Taylor 2ης τάξης

Από τον τύπο (2.4) για n=2 προκύπτει:

$T^{(2)}(t_i , w_i) = f(t_i , w_i) + \frac{h}{2} f'(t_i , w_i)$

Και σε συνδυασμό με τους τύπους (2.3) προκύπτει η σχέση:

$w_0=\alpha$

$w_{(i+1)} = w_i + h\, T^{(2)}(t_i , w_i)$ i=0,1,…,N-1

Με:

$T^{(2)}(t_i , w_i) = f(t_i w_i) + \frac{h}{2} f'(t_i,w_i)$

που ονομάζεται μέθοδος του Taylor τάξης 2.

Μέθοδος Taylor 4ης τάξης

Από τον τύπο (2.4) για n=4 προκύπτει:

$T^{(4)}(t_i ,w_i) = f(t_i, w_i) + \frac{h}{2} f' (t_i,w_i) + \frac{h^2}{6} f''(t_i.w_i) + \frac{h^3}{24} f'''(t_i,w_i)$

Και σε συνδυασμό με τους τύπους (2.3) προκύπτει η σχέση:

$w_0=\alpha$

$w_{i+1} = w_i + hT^4(t_i,w_i)$

Όπου

$T^{(4)}(t_i,w_i) = f(t_i,w_i) + \frac{h}{2} f'(t_i,w_i) + \frac{h^2}{6} f''(t_i,w_i) + \frac{h^3}{24} f'''(t_i,w_i)$

Παράδειγμα 2.1:

Εφαρμόζοντας την μέθοδο Taylor $2^{ης}$ τάξης και την μέθοδο Taylor $4^{ης}$ τάξης στο πρόβλημα αρχικών τιμών

$$y'=-y+t+1 \quad , 0 \le t \le 1 \quad ,y(0)=1$$

Λύση:

Για να εφαρμοστούν οι παραπάνω μέθοδοι πρέπει να βρεθούν οι τρεις πρώτοι παράγωγοι της συνάρτησης

$f(t,y(t)) = -y+t+1$

$f'(t,y(t)) = \frac{d}{dt}(-y+t+1)= -y'+1=-(-y+t+1)+1=y-t-1+1=y-t$

$f''(t,y(t))= \frac{d}{dt}(y-t)=y'-1=-y+t+1-1=-y+t$

$f'''(t,y(t))= \frac{d}{dt}(-y+t)=-y'+1=-(-y+t+1)+1=y-t+1-1=y-t$

επιπλέον,

$T^{(2)}(t_i, w_i)=f(t_i, w_i) + \frac{h}{2}f'(t_i , w_i)$

$= - w_i + t_i +1+\frac{h}{2}(w_i-t_i)$

$=(1-\frac{h}{2})(t_i-w_i)+1$

και

$T^{(4)}(t_i,w_i)=f(t_i, w_i)+ \frac{h}{2} f'(t_i,w_i)+ \frac{h^2}{6} f''(t_i,w_i)+ \frac{h^3}{24} f'''(t_i,w_i)$

$=-w_i+t_i +1+\frac{h}{2}(w_i-t_i) + \frac{h^2}{6}(-w_i+t_i) + \frac{h^3}{24}(w_i-t_i)$

$=(1-\frac{h}{2} + \frac{h^2}{6} - \frac{h^3}{24})(t_i-w_i)+1$

Μέθοδος Taylor 2ης τάξης:

Στους τύπους (2.5) εφαρμόζω τα δεδομένα της συγκεκριμένης άσκησης

$w_o=1$

$w_{i+1}=w_i+h[(1-h/2)(t_i-w_i)+1]$ για i=0,1,…,N-1

Χρησιμοποιώντας h=o.1 , N=10 και t_i =0,1 για κάθε i=1,2,…,0

Προκύπτει :

$w_0=1$

$w_{i+1}=w_i+0.1[(1-\frac{1}{2})(0,1i - w_i)+1]$

$= 0,905w_i + 0,0095i+0,1$

<u>Μέθοδος Taylor 4ης τάξης</u>

Στους τύπους (2.6) εφαρμόζοντας τα δεδομένα της συγκεκριμένης άσκησης προκύπτει:

$w_0=1$

$w_{i+1}=wi + h[(1- \frac{h}{2}+\frac{h^2}{6} -\frac{h^3}{24})(t_i - w_i)+1]$

Χρησιμοποιώντας τις προαναφέρθηκες τιμές h,N,ti προκύπτει:

$w_0=1$

$w_{i+1}=w_i + 0,1[(1- \frac{0.1}{2}+\frac{0.01}{6}-\frac{0.001}{24})(0,1i-w_i)+1]$

$=0,9048375w_i + 0,00951625i + 0,1$

για κάθε i=1,2,…,10

Στον πίνακα που ακολουθεί, παρουσιάζονται τα αποτελέσματα από την επίλυση του προβλήματος αυτού, με τα προγράμματα Taylor Order2 και Taylor Order4 που έχει κάνει η συγγραφέας της διπλωματικής αυτής σε Fortran.Επιπλέον στον πίνακα υπάρχουν οι ακριβείς λύσεις για κάθε t_i ,i=0,1,…,10 υπολογισμένες από την ακριβής λύση της διαφορικής εξίσωσης που εύκολα αποδεικνύεται ότι είναι η εξίσωση $y(t)=t + e^{-t}$

Στην τέταρτη και στην έκτη στήλη εμφανίζονται τα τοπικά σφάλματα αποκοπής των δύο μεθόδων σε κάθε κομβικό σημείο. Τα προγράμματα παρουσιάζονται αναλυτικά στο παράρτημα.

κομβικά σημεία	**προσεγγιστικές λύσεις με Taylor2**	**Ακριβείς λύσεις yi**	**Taylor2 Error**	**προσεγγιστικές λύσεις με Taylor4**	**Taylor4Error**
0	1	1	0	1	0
0.1	1,005	1,004837	0,000163	1,004883	0
0.2	1.019025	1,018731	0,000294	1,018814	0,000083
0.3	1,041218	1,040818	0,000400	1,040931	0,000113
0.4	1,070802	1,07032	0,000482	1,070456	0,000136
0.5	1.107076	1,106531	0,000545	1,106680	0,000149
0.6	1.149403	1,148812	0,000591	1,148979	0,000169
0.7	1,19721	1,196585	0,000625	1,196762	0,000177
0.8	1,249975	1,249329	0,000646	1,249511	0,000182
0.9	1,307227	1,30657	0,000657	1,306755	0,000185
1	1,368541	1,367879	0,000662	1,368066	0,000187

■

Παράδειγμα 2.2

Ένα βλήμα μάζας m=0,11 kg εκτοξεύεται με όπλο κάθετα προς τα επάνω με αρχική ταχύτητα (0)=8 meter/sec.Επιβραδύνεται εξ αιτίας της δύναμης της βαρύτητας

F_b =-mg και της αντίστασης του αέρα $Fr=-kg^2$ όπου g=9.8 meter/sec^2

και k=0,002 kg/meter είναι η σταθερά της. Είναι η επιτάχυνση της βαρύτητας. Η διαφορική εξίσωση που δίνει την ταχύτητα του βλήματος είναι:

$$mu = -mg – ku^2 \qquad (2.8)$$

Να υπολογιστεί η ταχύτητα του βλήματος τις χρονικές στιγμές 0.1 , 0.2, …, 1 seconds χρησιμοποιώντας την μέθοδο Taylor $2^{ης}$ τάξης

Λύση

Η διαφορική εξίσωση (2.8) μπορεί να γραφτεί ισοδύναμα

$$u' = -g - k/mu^2$$

Ισχύει ότι: $u'' = \frac{d}{dt}(u) = \frac{d}{dt}(-g-\frac{k}{m}u^2) = k/m2u.u^2$

$$=-2\frac{k}{m}u(-g-\frac{k}{m}u^2)=$$

$$= +2\frac{k}{m}gu + 2(\frac{k}{m})^2 u^3$$

Τότε

$$T^{(2)}(u_i) = -g-\frac{k}{m}u_1^2 + h/2(2\frac{k}{m}\, g\, u_i + 2(\frac{k}{m})^2 u_i^3)$$

$$= -g -\frac{k}{m}u_i^2 + h\frac{k}{m}(gu_i + \frac{k}{m}u_i^3)$$

Εφαρμόζοντας την μέθοδο Taylor 2ης τάξης προκύπτουν τα παρακάτω αποτελέσματα. Ο πίνακας που ακολουθεί αποτελεί και από τις προσεγγιστικές τιμές της ταχύτητας καθώς και των δύο δυνάμεων, της βαρύτητας F_b και της απόστασης του αέρα Fr σε κάθε κομβικό σημείο ti i=1,10 με t_0= o sec, t_1= 0,1sec, t2=o,2sec,…,t_{10} =1sec όπως προκύπτουν από το πρόγραμμα Vertical Shot Taylor 2 που έχει κάνει η συγγραφέας της διπλωματικής σε Fortran. Το πρόγραμμα παρουσιάζεται αναλυτικά παράρτημα.

κομβικά σημεία	**ταχύτητα**	**Βαρύτητα**	**Αντίσταση του αέρα**
0	8	-1,078	
0,1	7	-1,078	$-9{,}84*10^5$
0,2	6,042481	-1,078	$-7{,}30*10^5$
0,3	5,068176	-1,078	$-5{,}14*10^5$
0,4	4,092045	-1,078	$-3.34896*10^{-2}$
0,5	3,114047	-1,078	$-1{,}94*10^5$
0,6	2,134144	-1,078	$-9{,}11*10^4$
0,7	1,152295	-1,078	$-2{,}66*10^4$
0,8	0,1684587	-1,078	$-5{,}68*10^2$

0,9	-8174044	-1,078	$-1,34*10^4$
1	-1,805336	-1,078	$-6,52*10^4$

Από τον πίνακα προκύπτει ότι για t=0,8sec το βλήμα βρίσκεται στο πιο υψηλό σημείο, με ταχύτητα σχεδόν μηδενική, καθώς εκτελεί ευθύγραμμη ομαλά επιβραδυνόμενη κίνηση όπου θετική φορά θεωρήθηκε η φορά προς τα επάνω. Από t=0,8 εώς t=1sec το μέτρο της ταχύτητας είναι αρνητικό, δηλ το βλήμα αλλάζει φορά και πλέον πέφτει με την ταχύτητα του να παρουσιάζει άνοδο όσο το βλήμα πλησιάζει στο έδαφος. Το μέτρο της βαρυτικής δύναμης F_b παραμένει σταθερό καθώς είναι ανεξάρτητη από την ταχύτητα του βλήματος, ενώ η αντίσταση του αέρα Fr παίρνει διάφορες τιμές κάθε χρονική στιγμή με μέγιστη την στιγμή t_0= 0 όπου έχουμε και την μέγιστη ταχύτητα που έχει αρνητικό πρόσημο καθώς είτε το βλήμα ανεβαίνει είτε κατεβαίνει η αντίσταση του αέρα είναι δύναμη με φορά αντίθετη από εκείνη της ταχύτητας.■

Στο σημείο αυτό παρουσιάζονται οι αλγόριθμοι των μεθόδων Taylor $2^{ης}$ τάξης και Taylor $4^{ης}$ τάξης του δίνονται από τους τύπους (2.5) και (2.6) έτσι όπως κατασκευάστηκαν από την συγγραφέα της διπλωματικής.

<u>Αλγόριθμος Taylor $2^{ης}$ τάξης</u>

- Εισαγωγή των άκρων του διαστήματος [0,0] και του πλήθους των διαστημάτων N.
- Προσδιορισμός της αρχικής συνθήκης w=a
- Υπολογισμός του βήματος η από τον τύπο h=(b-a)N
- Προσδιορισμός του πρώτου κομβικού σημείου t=a
- Υπολογισμός της πρώτης παραγώγου της f
- Καθώς (Do)
- i=0,1,…,N-1
- $T_2(ti,wi)=f(ti,wi)+ h/2\ f\ (ti,wi)$
- $W=wi + h\ T_2(ti,wi)$
- T=t+h
- Εκτύπωση των (τ,w)
- Τέλος

Αλγόριθμος Taylor $4^{ης}$ τάξης

- Εισαγωγή των άκρων του διαστήματος [a,b] και του πλήθους των διαστημάτων N.
- Προσδιορισμός της αρχικής συνθήκης w=a
- Υπολογισμός του βήματος h από τον τύπο h=(b-a)/N
- Προσδιορισμός του πρώτου κομβικού σημείου t=a
- Υπολογισμός των τριών πρώτων παραγώγων της f
- Καθώς (Do) i=0,1,…,N-1
  $T_4(ti,wi)=f(ti,wi)+h/2\ f'(ti,wi) + h^2/6\ f''(ti,wi) + h^3/24\ f'''(ti,wi)$
  $w_{i+1}=w_i+h\ T_4(ti,wi)$
  T=t+h
- Εκτύπωση των (t,w)
- Τέλος

2.3 Μέθοδοι Runge Kutta

2.3.1 Εισαγωγικές έννοιες

Το πρόβλημα αυτό ανήκει στην κατηγορία μεθόδων απλού βήματος. Η μέθοδος του Euler που εξετάστηκε είναι απλή στην εφαρμογή της, όμως δεν είναι κατάλληλη για την πράξη, καθώς έχει μικρή ακρίβεια. Για καλή ακρίβεια απαιτείται ένα κατάλληλο μικρό βήμα h.

Η μέθοδος του Taylor δίνει ασφαλώς καλύτερα αποτελέσματα τα οποία συγκλίνουν περισσότερο στην ακριβή κλίση, όσο μεγαλύτερη είναι η τάξη της μεθόδου. Το μειονέκτημα των μεθόδων Taylor είναι ότι απαιτείται ο υπολογισμός ολικών παράγων ανώτερης τάξης, πράγμα που έχει ως αποτέλεσμα την αύξηση του πλήθους των πράξεων.

Ο Runge (1895) και αργότερα ο Kutta (1901) έδειξαν ότι μπορούμε να δώσουμε μεθόδους αριθμητικής επίλυσης προβλημάτων αρχικών τιμών της παραπάνω μορφής, απλού βήματος με εξίσου καλή ακρίβεια με την μέθοδο Taylor, που δεν απαιτούν τον υπολογισμό παραγόντων. Οι μέθοδοι αυτοί ονομάζονται, Runge και Kutta. Όλες οι μέθοδοι Runge και Kutta έχουν την μορφή :

$$y_{i+1} = y_i + h\varphi\ (x_1, y_1)$$

Όπου συνάρτηση φ είναι μία κατάλληλη προσέγγιση της f(x,y) στο διάστημα [xi,xi+1]. Υπολογίζοντας την φ προκύπτουν οι μέθοδοι Runge- Kutta.

2.3.2 Η οικογένεια των μεθόδων Runge- Kutta 2ης τάξης.

Ο υπολογισμός της φ για την μέθοδο δεύτερης τάξης Runge και Kutta γίνεται ως εξής:

Στο διάστημα [xi,xi+1] έχουμε $\varphi=\alpha\kappa_1+bk_2$

$y(i) + h (ak_1+bk_2)$

Αν πάρουμε

$K_1=f(x_1,y_1)$

$K_2=f(x_1+ph,yi+ghf(x_1,y_1))=f(x_1+ph,y_1+ghk_1)$

Και αν αναπτύξουμε την K_2 κατά Taylor έχουμε

$K_2=f(x_1+ph,y_1+ghf(x_1,y_1))=f(x_1,y_1)+p\ h\ f_x(x_1,y_1) +g\ h\ f(x_1,y_1)\ f_y(x_1,y_1) + O(h^2)$

Με αντικατάσταση έχουμε:

$$y_{i+1}=y_i+h[af(x_1,y_1)+bf(x_1,y_1)]+h^2[bpf_x(x_1,y_1)+bqf(x_1,y_1)\ f_y(x_1,y_1)]+O(h^3)$$

Αν αναπτύξουμε την y(x) κατά Taylor έχουμε

$$y_{i+1}=h(x_1+h)=y(x_1)+hf(x_1,y(x_1))+\frac{h^2}{2!}\ f'(x_1,\ y(x_1)) +\frac{h^3}{3!}\ f''(\xi,\ y(\xi))$$

$$y\ (x_1+h)=y_{i+1}=y(x_1)+hf(x_1.y(x_1))+\frac{h^2}{2!}\ f'(x_1,\ y(x_1)) +\frac{h^3}{3!}\ f''(\xi,\ y(\xi))$$

Όπου, $f=F_X + Fy$

Αν υποθέσουμε ότι οι συντελεστές των δυνάμεων του h στις δύο μορφές της y_{i+1} ,με εξίσωση των όμοιων όρων προκύπτει:

$\alpha+b=1$

$bp=\frac{1}{2}$

$bq=\frac{1}{2}$

Οι τρείς εξισώσεις περιέχουν τέσσερεις αγνώστους από τους οποίους ο ένας μπορεί να οριστεί αυθαίρετα.

Αν πάρουμε $b=\frac{1}{2}$ προκύπτει:

$p=q=\frac{1}{2}$

$a=\frac{1}{2}$

Ο τύπος τότε γίνεται:

$y_{i+1}=y_i+hf(x_i + \frac{h}{2} , y_i+ \frac{h}{2} f(x_i ,y_i)$ (2.10)

Ο τύπος αυτός είναι γνωστός ως βελτιωμένη μέθοδος του Euler (Modified Euler).

Το πρώτο βήμα στην δημιουργία των μεθόδων Runge- Kutta είναι ο προσδιορισμός των τιμών a_1, b_1, c_1 με την προυπόθεση ότι $a_1f(t+b_1, y+c_1)$ προσεγγίζει το πολυώνυμο Taylor δευτέρου βαθμού

$T^{(2)}(t,y)= f(t,y) + \frac{h}{2} f'(t,y)$

Με σφάλμα μικρότερο από $O(h^2)$, που είναι το σφάλμα αποκοπής της μεθόδου Taylor $2^{ης}$ τάξης. Καθώς,

$f'(t,y)= \frac{df}{dt}(t,y)= \frac{df}{dt}(t,y) + \frac{df}{dy}(t,y)y'(t)$

$y'(t)= f(t,y)$

Επομένως,

$T^{(2)}(t,y)= f(t,y) + \frac{h}{2}\frac{df}{dt}(t,y) + \frac{h}{2}\frac{df}{dy}(t,y)f'(t,y)$ (2.11)

Αναπτύσσοντας την $f(t+b_1,y+c_1)$ στο πολυώνυμο Taylor $1^{ου}$ βαθμού για (t,y) προκύπτει:

$a_1f(t+b_1, y+c_1)=a_1 f(t,y) + a_1b_1\frac{df}{dt}(t,y)+ a_1c_1\frac{df}{dy}(t,y) + a_1R_1(t+b_1,y+c_1)$

Εξισώνοντας τους συντελεστές της f και των παραγώγων της στις σχέσεις (2.10) και (2.11) προκύπτουν τρεις εξισώσεις :

$f(t,y)$: $a_1=1$

$\frac{df}{dt}(t,y)$: $a_1b_1=\frac{h}{2}$

$\frac{df}{dy}(t,y)$: $a_1c_1=\frac{h}{2}f(t,y)$

Επομένως,

$T^{(2)}(t,y) = f(t+\frac{h}{2}, y+\frac{h}{2} f(t,y)) - R_1(t+\frac{h}{2}, y+\frac{h}{2} f(t,y))$.

Η διαφορική μέθοδος που προκύπτει αντικαθιστώντας το $T^{(2)}(t,y)$ στην μέθοδο του Taylor $2^{ης}$ τάξης από την $f(t+\frac{h}{2}, y+\frac{h}{2} f(t,y))$ είναι μια ειδική μορφή μεθόδου Runge – Kutta που είναι γνωστή ως Midpoint Method.

Μέθοδος Midpoint

$w_o = a$

$w_{i+1} = w_i + hf(t_1 + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i))$ (2.13)

για κάθε i = 0,1 ,……..., N- 1

Το σφάλμα αποκοπής (τοπικά) είναι της μορφής $0(h^2)$. Στους τύπους (2.9) επιλέγοντας $\alpha = \frac{1}{4}$ προκύπτει :

$b = \frac{3}{4}$

$p = q = \frac{2}{3}$

Η διαφορική εξίσωση που προκύπτει για τις παραπάνω τιμές των σταθερών α, b, p και q είναι η εξής

$w_o = \alpha$ (2.14)

$w_{i+1} = w_i + \frac{h}{4} [f(t_i, w_i) + 3 f(t_i + \frac{2}{3} h, w_i + \frac{2}{3} h f(t_i, w_i)]$

και ονομάζεται μέθοδος Heun.

Όλες αυτές οι μέθοδοι, κατατάσσονται στις μεθόδους Runge – Kutta $2^{ης}$ τάξης. Το σφάλμα αποκοπής είναι και στις τρεις μεθόδους τάξης $O(h^2)$. Στην συνέχεια παραθέτονται οι αλγόριθμοι των τριών μεθόδων που έχουν πολλά κοινά σημεία μεταξύ τους, όπως παραπλήσιος ήταν και ο τρόπος παραγωγής των μεθόδων.

Αλγόριθμος για την Μέθοδο Midpoint

- Εισαγωγή των άκρων του διαστήματος [a, b] και του πλήθους των διαστημάτων N
- Προσδιορισμός της αρχικής συνθήκης

  $w = w_0$

- Υπολογισμός του βήματος h από τη σχέση

  h= (b - a) / N

- Προσδιορισμός του πρώτου σημείου t=α, δηλαδή με το αριστερό άκρο του διαστήματος
- Καθώς (D_o) i= 0,1,……., N-1

  w= w+ h f (t+ $\frac{h}{2}$, w +$\frac{h}{2}$ f (t, w))

  t = t + h

- Εκτύπωση των (t, w)
- Τέλος

Αλγόριθμος για την μέθοδο Heun

- Εισαγωγή των άκρων του διαστήματος [α, b] και του πλήθους των διαστημάτων N
- Προσδιορισμός της αρχικής συνθήκης
- w =w_0
- Υπολογισμός του βήματος h από την σχέση
- h = (b – α) /N
- Προσδιορισμός του πρώτου κομβικού σημείου t=α
- Καθώς (D_o) i= 0,1,……,N-1

  w=w + $\frac{h}{4}$ [f (t, w) + 3 f (t + $\frac{2}{3}$ h , w + $\frac{2}{3}$ h f (t , w))]

  t = t + h
- Εκτύπωση των (t,w)
- Τέλος

Αλγόριθμος για την μέθοδο Modified Euler

- Εισαγωγή των άκρων του διαστήματος [α, b] και του πλήθους των διαστημάτων N
- Προσδιορισμός της αρχικής συνθήκης

  w= w_0
- Υπολογισμός του βήματος h από την σχέση

  h = (b –α) /N
- Προσδιορισμός του πρώτου κομβικού σημείου t=a
- Καθώς (D_o) i = 0,1,……, N-1

  w = w + $\frac{h}{2}$ [f (t , w) + f (t + h, w + h f (t, w))]

  t= t+ h
- Εκτύπωση των (t, w)
- Τέλος

Παράδειγμα 2.3:

Να εφαρμοστεί κάθε μία από τις μεθόδους Runge – Kutta $2^{ης}$ τάξης στην διαφορική εξίσωση

$y' = - y+t +1 \quad 0 \leq t \leq 1 \quad y(0) = 1$

Λύση :

Παρατηρείται ότι όποια μέθοδο και αν επιλεχθεί για την λύση της διαφορικής εξίσωσης καταλήγουμε στην εξίσωση

$w_o = 1$

$w_{i+1} = 0,905\, w_i + 0,0095i + 0,1$

Αυτό οφείλεται στην φυσική και στα χαρακτηριστικά της συγκεκριμένης εξίσωσης. Στην συνέχεια αποδεικνύεται χρησιμοποιώντας τους τύπους (2.12), (2.13) και (2.14) οι τρείς μέθοδοι ταυτίζονται.

Οι μέθοδοι Modified Euler, Midpoint και Heun έχουν την ίδια αρχική συνθήκη και όλες λαμβάνουν χώρα για i= 0,1,….,N-1. Επομένως για να αποδειχθεί ότι ταυτίζονται, αρκεί να αποδειχθεί ότι οι τρεις τύποι

$w_{i+1} = w_i + \frac{h}{2}[f(t_i, w_i) + f (t_i + 1, w_i + h f (t_i, w_i))]$

$w_{i+1} = w_i + h \left[f\left(t_i + \frac{h}{2} , w_i + \frac{h}{2} f (t_i, w_i)\right) \right]$

$w_{i+1} = w_i + \frac{h}{4}[f (t_i , w_i) + 3f (t_i + \frac{2}{3} h , w_i + \frac{2}{3} h f(t_i, w_i))]$

Ταυτίζονται σε κάθε επανάληψη της κάθε μεθόδου. Επομένως αρκεί να δεχτεί ότι

$\frac{h}{2}[f (t,w) + f(t, w + h f(t, w))] =$

$h \left[f \left(t + \frac{h}{2} , w + \frac{h}{2} f (t,w)\right)\right] =$

$\frac{h}{4} [f(t , w) + 3 f\left(t + \frac{2}{3} h, w + \frac{2}{3} h f (t ,w)\right)]$

Ισχύει ότι:

$\frac{h}{2}[f (t,w) + f(t, w+h f(t, w))] =$

$\frac{h}{2}$[-w + t + 1- w- hf(t ,w) + t+h+1]=

$\frac{h}{2}$[- w + t + 1 – w - h(-w + t + 1) + t + h +1]=

$\frac{h}{2}$[-2w + 2t + 2 + hw – ht – h + h]=

h(-w + t +1 + $\frac{hw}{2}$ - $\frac{ht}{2}$).

Επίσης ισχύει:

$\frac{h}{4}$ [f(t,w) +3 f(t + $\frac{2}{3}$ h, w+ $\frac{2}{3}$ hf(t,w))]=

$\frac{h}{4}$ [f(t,w) +3 (-w- $\frac{2}{3}$h f(t,w)+t+ $\frac{2}{3}$h +1)]=

$\frac{h}{4}$ [f(t,w) -3 w – 2hf(t,w) +3t+2h+3]=

$\frac{h}{4}$ [-w+t+1-3w-2h(-w+t+1)+3t+2h+3]=

$\frac{h}{4}$ (-4w+4t+4+2hw-2ht-2h+2h)=

h(-w+t+1+ $\frac{hw}{2}$ - $\frac{ht}{2}$

Ομοίως ισχύει:

h [f (t+ $\frac{h}{2}$, w + $\frac{h}{2}$ f (t,w))] =

h[-w - $\frac{h}{2}$ f(t,w) + t+ $\frac{h}{2}$ +1]=

h[-w - $\frac{h}{2}$(-w+t+1) +t + $\frac{h}{2}$ +1]=

h(-w + $\frac{hw}{2}$ - $\frac{ht}{2}$ - $\frac{h}{2}$ + t + $\frac{h}{2}$+1)=

h(-w + t+1 + $\frac{hw}{2}$ - $\frac{ht}{2}$)

Επομένως είναι ίσα μεταξύ τους. ■

Παράδειγμα 2.4

Να λυθεί η κανονική διαφορική εξίσωση

$$y = -y + t^2 + 1, \qquad 0 \le t \le 1 \qquad y(0) = 1$$

με τις τρεις μεθόδους Runge – Kutta $2^{ης}$ τάξης και να συγκριθούν τα αποτελέσματα.

Λύση:

Η ακριβής λύση της εξίσωσης εύκολα αποδεικνύεται ότι είναι:

$y = -2e^{-t} + t^2 - 2t + 3$

Στον πίνακα που ακολουθεί εμφανίζονται οι προσεγγιστικές τιμές των τριών μεθόδων σε κάθε κομβικό σημείο. Επιπλέον, υπάρχουν στις τρεις τελευταίες στήλες τα αντίστοιχα σφάλματα αποκοπής. Οι προσεγγιστικές τιμές υπολογίστηκαν με τα προγράμματα MidpointMethod, ModifiedEulerMethod και HeunMethod σε Fortran που δημιούργησε η συγγραφέας της διπλωματικής και παρουσιάζονται αναλυτικά στο παράρτημα.

Κομβικά σημεία	Ακριβείς λύσεις	Modified Euler	Heun Method	Midpoint	Modified Euler Error	Heun Method Error	Midpoint Method Error
0	1	1.000000	1.000000	1.000000	0	0	0
0.1	1,004837	1.005000	1.010000	1.005000	0.000163	0.005163	0.000163
0.2	1,018731	1.019025	1.029000	1.019025	0.000294	0.000294	0.000294
0.3	1,040818	1.041218	1.056100	1.041218	0.00040	0.015282	0.0004
0.4	1,07032	1.070802	1.090490	1.070802	0.000482	0.02058	0.000482
0.5	1,106531	1.107076	1.131441	1.107076	0.000545	0.02491	0.000545
0.6	1,148812	1.149403	1.178297	1.149403	0.000591	0.029485	0.000591
0.7	1,196585	1.197210	1.230467	1.197210	0.000625	0.033882	0.000625

0.8	1,249329	1.249975	1.287421	1.249975	0.000646	0.038092	0.000646
0.9	1,30657	1.307227	1.348678	1.307227	0.00113	0.042108	0.000657
1	1,367879	1.368541	1.413811	1.368541	0.000662	0.045932	0.000662

■

2.3.3 Η μέθοδος Runge – Kutta 3ης τάξης

Η μέθοδος τρίτης τάξης Runge – Kutta προκύπτει όπως και οι μέθοδοι δεύτερης τάξης. Θεωρούμε,

$$\varphi = ak_1 + bk_2 + ck_3 \qquad (2.14)$$

όπου τα k_1, k_2, k_3 ορίζονται από τις σχέσεις:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + \rho h, y_i + qhk_1) \qquad (2.15)$$

$$k_3 = f(x_i + rh, y_i + shk_1 + thk_2)$$

Οι σταθερές a, b, c, ρ, r, t, s υπολογίζονται, αναπτύσσοντας κατά Taylor τα k_2 και k_3 ως συναρτήσεις δύο μεταβλητών. Στη συνέχεια αναπτύσσεται και η συνάρτηση y (x) κατά Taylor. Εξισώνοντας τους συντελεστές των ομοβάθμιων όρων του h μέχρι h^3 προκύπτουν οι εξισώσεις:

$$a + b + c = 1$$

$$b\rho + cr = \frac{1}{2}$$

$$b\rho^2 + cr^2 = \frac{1}{3}$$

cρs = $\frac{1}{6}$

ρ = q

r = s + t

Το παραπάνω σύστημα αποτελείται από 6 εξισώσεις οκτώ αγνώστων. Επομένως δύο από τους αγνώστους μπορούν να ορισθούν αυθαίρετα.

Επιλέγοντας ότι a = 1/6 και c = 1/6 προκύπτει από το σύστημα ότι:

b =$\frac{4}{6}$, ρ = q =$\frac{1}{2}$, r = 1, s = -1, t = 2

Αντικαθιστώντας τις παραμέτρους στις σχέσεις (2.15) και εν συνεχεία στη (2.14) προκύπτει ο τύπος

$y_{i+1} = y_i + h/6 (k_1 + 4k_2 + k_3)$ (2.16)

$k_1 = hf\ x_i, y_i)$

$k_2 = hf (h_i + \frac{1}{2}h, y_i + \frac{1}{2} k_1)$

$k_3 = hf (x_i + h, y_i – k_1 + 2k_2)$

ο οποίος μπορεί να γραφτεί ισοδύναμα ως εξής

$y_{i+1} = y_i + \frac{h}{6}(k_1 + 4k_2 + k_3)$

$k_1 = f(x_i, y_i)$

(2.17)

$k_2 = hf(h_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)$

$k_3 = hf(x_i + h, y_i - hk_1 + 2hk_2)$

Αλγόριθμος Runge – Kutta 3ης τάξης

- Εισαγωγή των άκρων του διαστήματος [a, b] και του πλήθους N
- Προσδιορισμός της αρχικής συνθήκης
  $w = w_0$
- Υπολογισμός του βήματος h από την σχέση
  $h = (b - a) / N$
- Προσδιορισμός του πρώτου κομβικού σημείου t = a
- Καθώς (D_0) i = 0, 1, …, N υπολογισμός των
  $k_1 = f(t, w)$
  $k_2 = hf(t + \frac{1}{2}h, w + \frac{1}{2}k1)$
  $k_3 = hf(t + h, w - k_1 + 2k_2)$
  $w = w + \frac{1}{6}h(k_1 + 4k_2 + k_3)$
  $t = t + h$
- Εκτύπωση των (t, w)
- Τέλος

Παράδειγμα 2.4

Να εφαρμοστεί η μέθοδος Runge – Kutta 3ης τάξης στο πρόβλημα αρχικών τιμών:

$$y' = -y + t + 1, \qquad 0 \le t \le 1, \qquad y(0) = 1$$

Λύση:

κομβικά σημεία	ακριβης λύση	προσεγγιστική λύση	Σφάλμα RungeKutta3
0	1	1	0

0,1	1,004837	1.004833	0.000004
0,2	1,018731	1.018723	0.000008
0,3	1,040818	1.040808	0.00001
0,4	1,07032	1.070308	0.000012
0,5	1,106531	1.106517	0.000014
0,6	1,148812	1.148797	0.000015
0,7	1,196585	1.196570	0.000015
0,8	1,249329	1.249313	0.000016
0,9	1,30657	1.306553	0.000017
1	1,367879	1.367863	0.000016

Στον πίνακα παρουσιάζονται οι ακριβείς και οι προσεγγιστικές λύσεις σε κάθε κομβικό σημείο, καθώς και τα σφάλματα αποκοπής. Οι προσεγγιστικές τιμές είναι τα αποτελέσματα του προγράμματος RungeKuttaOrder3 που δημιούργησε η συγγραφέας της διπλωματικής σε Fortran. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα.■

2.3.4 Η οικογένεια των μεθόδων Runge – Kutta 4ης τάξης

Η μέθοδος τέταρτης τάξης Runge – Kutta προκύπτει όπως οι προηγούμενες μέθοδοι. Η απόδειξη του τύπου Runge – Kutta που ακολουθεί έγινε από τη συγγραφέα της διπλωματικής.

Απόδειξη 2.1

Θεωρούμε τη συνάρτηση

$\varphi = ak_1 + bk_2 + ck_3 + dk_4$ όπου

$k_1 = hf\ (x, y)$

$k_2 = hf\ (x + mh, y + mk_1)$ (2.18)

$k_3 = hf\ (x + nh, y + nk_2)$

$k_4 = hf\ (x + \rho h, y + \rho k_3)$

όπου $y(x+h) - y(x) \sim \varphi$

Παραγωγίζουμε την $y' = f(x, y)$

$$y^{(2)} = f_x + f_y y' = f_x + f_y f = F_1$$

$$y^{(3)} = f_{xx} + 2ff_{xy} + f^2 f_{yy} + f_y (f_x + ff_y) = F_2 + f_y F_1$$

$$y^{(4)} = f_{xxx} + 3ff_{xxy} + 3f^2 f_{xyy} + f^3 f_{yyy} + f_y (f_{xx} + 2ff_{xy} + f^2 f_{yy}) + 3(f_x + ff_y)(f_{xy} + ff_{yy}) + f_y^2 (f_x + ff_y) = F_3 + f_y F_2 + 3F_1 (f_{xy} + ff_{yy}) + f_y^2 F_1$$

όπου για λόγους ευκολίας ορίσαμε

$$F_1 = f_x + ff_y$$

$$F_2 = f_{xx} + 2ff_{xy} + f^2 f_{yy} \qquad (2.19)$$

$$F_3 = f_{xxx} + 3ff_{xxy} + 3f^2 f_{xyy} + f^3 f_{yyy}$$

Επομένως η σειρά Taylor, εώς τον όρο h^4 γράφεται:

$$y(x+h) - y(x) = hf + \frac{1}{2} h^2 F_1 + \frac{1}{6} h^3 (F_2 + f_y F_1) + \frac{1}{24} h^4 [F_3 + f_y F_2 + 3(f_{xy} + ff_{yy}) F_1 + f_y^2 F_1]$$

Αν αναπτύξουμε τα k_2, k_3, k_4 ως σειρά Taylor προκύπτουν

$$k_2 = h [f + mhF_1 + \frac{1}{2} m^2 h^2 F_2 + \frac{1}{6} m^3 h^3 F_3]$$

$$k_3 = h [f + nhF_1 + \frac{1}{2} h^2 (n^2 F_2 + 2mnf_y F_1) + \frac{1}{6} h^3 (n^3 F_3 + 3m^2 nf_y F_2 + 6mn^2 (f_{xy} + ff_{yy}) F_1]$$

$k_4 = h\,[f + \rho h F_1 + \frac{1}{2} h^2 (\rho^2 F_2 + 2n\rho f_y F_1) + \frac{1}{6} h^3 (\rho^3 F_3 + 3n^2 \rho f_y F_2 + 6n\rho^2 (f_{xy} + f f_{yy}) F_1 + 6mn\rho f_y^2 F_1)]$

όπου έχουν απαλοιφθεί οι όροι μεγαλύτερου του h^3 καθώς πολλαπλασιασμένοι επί h θα προέκυπταν όροι μεγαλύτεροι του h^4 που δεν θα χρησιμοποιηθούν. Επίσης παρουσιάζονται οι τελικές μορφές των k_2, k_3, k_4 μετά τις αντικαταστάσεις των τύπων (2.19) για λόγους συντομίας.

Αντικαθιστώντας τα k_1, k_2, k_3, k_4 στον τύπο της φ προκύπτει:

$\varphi = ak_1 + bk_2 + ck_3 + dk_4$

$= ahf + bh\,[f + mhF_1 + \frac{1}{2} m^2 h^2 F_2 + \frac{1}{6} m^3 h^3 F_3] + ch\,[f + nhF_1 + \frac{1}{2} h^2 (n^2 F_2 + 2mnf_y F_1) +$

$+ \frac{1}{6} h^3 (n^3 F_3 + 3m^2 n f_y F_2 + 6mn^2 (f_{xy} + f f_{yy}) F_1] + dh\,[f + \rho h F_1 + \frac{1}{2} h^2 (\rho^2 F_2 + 2n\rho f_y F_1)$

$+ \frac{1}{6} h^3 (\rho^3 F_3 + 3n^2 \rho f_y F_2 + 6n\rho^2 (f_{xy} + f f_{yy}) F_1 + 6mn\rho f_y^2 F_1)]$

όπου εκτελώντας τις πράξεις και κάνοντας αναγωγή ως προς τους όρους h, h^2, h^3 και h^4 προκύπτει:

$\varphi = (a + b + c + d) hf + (bm + cn + d\rho) h^2 F_1 + \frac{1}{2} (bm^2 + cn^2 + d\rho^2) h^3 F_2 + \frac{1}{6} (bm^3 + cn^3 + d\rho^3) h^4 F_3 +$

$+ (cmn + dn\rho) h^3 f_y F_1 + \frac{1}{2} (cm^2 n + dn^2 \rho) h^4 f_y F_2 + (cmm^2 + dn\rho^2) h^4 (f_{xy} + f f_{yy}) F_1 +$
$dmn\rho h^4 f_y^2 F_1$ (2.20)

Η σειρά Taylor είναι:

$y(x + h) - y(x) = hf + \frac{1}{2} h^2 F_1 + \frac{1}{6} h^3 (F_3 + f_y F_1) + \frac{1}{24} h^4 [F_3 + f_y F_2 + 3(f_{xy} + f f_{yy}) F_1 + f_y^2 F_1]$ (2.21)

όμως $y(x + h) - y(x) \sim \varphi$, άρα συγκρίνοντας τους αντίστοιχους όρους των σχέσεων (2.20) και (2.21) προκύπτει το σύστημα

$a + b + c + d = 1$

$bm + cn + d\rho = \frac{1}{2}$

$bm^2 + cn^2 + d\rho^2 = \frac{1}{3}$

$bm^3 + cn^3 + d\rho^3 = \frac{1}{4}$

(2.22)

$cmn + dn\rho = \frac{1}{6}$

$cmm^2 + dn\rho^2 = \frac{1}{8}$

$cm^2n + dn^2\rho = \frac{1}{12}$

$dmn\rho = \frac{1}{24}$

Το σύστημα (2.22) αποτελείται από οκτώ εξισώσεις και επτά αγνώστους. Επιλύοντας το σύστημα προκύπτουν οι τιμές των σταθερών a, b, c, d, m, n, ρ ως εξής:

$a = d = \frac{1}{6}$, $b = c = \frac{1}{3}$, $m = n = \frac{1}{2}$ $\rho = 1$.

Αντικαθιστώντας τις τιμές αυτές στους τύπους (2.18) προκύπτει:

$k_1 = hf(x, y)$

$k_2 = hf(x + \frac{1}{2}h, y + \frac{1}{2}k_1)$

$k_3 = hf\left(x + \frac{1}{2}h,\ y + \frac{1}{2}k_2\right)$ (2.23)

$k_4 = hf\left(x + h,\ y + k_3\right)$

$\left(\varphi = \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)\right)$

$y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$ $i = 0, 1, \ldots, N-1$ ■

Ο τύπος (2.23) ονομάζεται μέθοδος Runge – Kutta $4^{ης}$ τάξης. Η μέθοδος Runge – Kutta όπως και όλες οι μέθοδοι $4^{ης}$ τάξης που θα αναφερθούν στη συνέχεια έχουν τοπικό σφάλμα αποκοπής $0(h^4)$. Η λύση y (t) της διαφορικής εξίσωσης έχει πέντε συνεχείς παραγώγους. Ο λόγος που εισάγεται η ορολογία των k1, k2, k3, k4 στη μέθοδο είναι για να εξαλειφθεί η ανάγκη της χρήσης πολλών παρενθέσεων στη δεύτερη μεταβλητή της συνάρτησης f (t, y).

Παρατήρηση 2.1:

Ο τύπος (2.23) Runge – Kutta αποδεικνύεται θεωρώντας

$k_1 = hf\left(x, y\right)$

$k_2 = hf\left(x + a_1h,\ y + b_1k_1\right)$

$k_3 = hf\left(x + a_2h,\ y + b_2k_1 + b_3k_2\right)$ (2.24)

$k_4 = hf\left(x + a_3h,\ y + b_4k_1 + b_5k_2 + b_6k_3\right)$

$y_{i+1} = y_i + w_1k_1 + w_2k_2 + w_3k_3 + w_4k_4$

Στην περίπτωση αυτή αντί του συστήματος (2.22) προκύπτει το σύστημα

$b_1 = a_1$

$b_2 + b_3 = a_2$

$b_4 + b_5 + b_6 = a_3$

$w_1 + w_2 + w_3 + w_4 = 1$

(2.25)

$w_2a_1 + w_3a_2 + w_4a_3 = \frac{1}{2}$

$w_2a_1^2 + w_3a_2^2 + w_4a_3^2 = \frac{1}{3}$

$w_2a_1^3 + w_3a_2^3 + w_4a_3^3 = \frac{1}{4}$

$w_3a_1b_3 + w_4 (a_1b_5 + a_2b_6) = \frac{1}{6}$

$w_3a_1^2b_3 + w_4 (a_1^2b_5 + a_2^2b_6) = \frac{1}{12}$

$w_4a_1b_3b_6 = \frac{1}{24}$

$w_3a_1a_2b_3 + w_4a_3 (a_1b_5 + a_2b_6) = \frac{1}{8}$

που αποτελείται από 11 εξισώσεις και 13 αγνώστους, 2 από τους οποίους ορίζονται αυθαίρετα. Επιλέγοντας $a_1 = \frac{1}{2}$, $b_2 = 0$ προκύπτει

$a_2 = \frac{1}{2}$, $a_3 = 1$

$b_1 = \frac{1}{2}$, $b_3 = b_4 = 0$, $b_5 = 0$, $b_6 = 1$

$w_1 = \frac{1}{6}$, $w_2 = \frac{1}{3}$, $w_3 = \frac{1}{3}$, $w_4 = \frac{1}{6}$

που μας δίνουν τη μέθοδο RungeKutta 4^{ης} τάξης αν αντικατασταθούν στη (2.24).

Ο συμβολισμός στη (2.24) δεν ακολουθήθηκε από τη συγγραφέα της διπλωματικής στην απόδειξη του τύπου RungeKutta για λόγους ευκολίας καθώς και επειδή το σύστημα είναι αρκετά πιο πολύπλοκο από το σύστημα (2.22).

Εκτός από τη μορφή (2.23) έχουν δοθεί και άλλοι τύποι.

Ο τύπος που ακολουθεί, έχει δοθεί από τον Kutta, και ονομάζεται μέθοδος Kutta.

$k_1 = f(x, y)$

$k_2 = f(x + \frac{1}{3}h, y + \frac{1}{3}h\, k_1)$

$k_3 = f(x + \frac{2}{3}h, y - \frac{1}{3}h\, k_1 + hk_2)$ (2.26)

$k_4 = f(x + h, y + hk_1 - hk_2 + hk_3)$

$y_{i+1} = y_i + \frac{1}{8}h\,(k_1 + 3k_2 + 3k_3 + k_4), \quad i = 0, 1, \ldots, N-1$

Η μορφή που χρησιμοποιείται περισσότερο σε προγράμματα ηλεκτρονικών υπολογιστών είναι του Gill, λέγεται Runge – Kutta – Gill και έχει τη μορφή:

$y_{i+1} = y_i + \frac{1}{3}h[k_1 + 2(1 - 1/\sqrt{2})\, k_2 + 2(1 + 1/\sqrt{2})\, k_3 + k_4]$

για $i = 0, 1, \ldots, N - 1$

όπου $k_1 = f(x_i, y_i)$

$k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)$

$$k_3 = f\left[x_i + \frac{1}{2}h,\ y_i + \left(-\frac{1}{2} + 1/\sqrt{2}\right) hk_1 + (1 - 1/\sqrt{2})\, hk_2\right] \qquad (2.27)$$

$$k_4 = f\left[x_i + h,\ y_i - 1/\sqrt{2}hk_2 + 1 + 1/\sqrt{2})\, hk_3\right]$$

Για την εκλογή του βήματος h πρέπει να υπάρχει και κάποιος υπολογισμός του σφάλματος καθώς λύνουμε τη διαφορική εξίσωση σε κάθε βήμα. Βασική προϋπόθεση για να έχουμε μικρό σφάλμα, πρέπει να διαλέξουμε μικρό βήμα h. Αν πάρουμε όμως πολύ μικρή τιμή του h, πρέπει να κάνουμε πολλούς υπολογισμούς, και αυξάνεται ο κίνδυνος να αναπτυχθεί το σφάλμα στρογγυλοποίησης, όπως ακόμα και να απαιτηθούν πολλές πράξεις, αφού για κάθε βήμα της μεθόδου απαιτείται υπολογισμός της παραγώγου m φορές.

Επιπλέον, μέθοδος RungeKutta πέρα από το μικρό σφάλμα της τάξης 0 (h^4) συγκλίνουν. Δηλαδή ισχύει πάντα

$$\lim_{h \to 0} (y_i - y(x_i)) = 0$$

Μια άλλη ιδιότητα των μεθόδων RungeKutta είναι η ευστάθεια της αριθμητικής λύσης. Αυτό σημαίνει ότι, αν για κάποιο λόγο δημιουργηθούν σφάλματα σε κάποιο στάδιο της λύσης, (όπως λανθασμένες τιμές στις αρχικές συνθήκες, τοπικό σφάλμα αποκοπής, σφάλμα στρογγυλοποίησης) η λύση δεν καταστρέφεται στα επόμενα βήματα.

Στη συνέχεια παρουσιάζονται οι αλγόριθμοι RungeKutta, Kutta και Gill που κατασκευάστηκαν από τη συγγραφέα της διπλωματικής. Από το σημείο αυτό οι αλγόριθμοι παρουσιάζονται με την μορφή βημάτων καθώς είναι πιο πολύπλοκοι και η μορφή που χρησιμοποιήθηκε έως τώρα δεν ενδείκνυται παρά μόνο για απλούς αλγορίθμους.

<u>Αλγόριθμος RungeKutta 4ης τάξης</u>

Βήμα 1 Εισαγωγή των άκρων a, b του [a, b]

Εισαγωγή του πλήθους N

Προσδιορισμός της αρχικής συνθήκης

$w = w_0$

Βήμα 2 Υπολογισμός του βήματος h

$h = (b - a) / N$

Προσδιορισμός του πρώτου κομβικού σημείου

$t = a$

Βήμα 3 Καθώς i = 0, 1, 2, …, N – 1

$k_1 = hf(t, w)$

$k_2 = hf(t + \frac{1}{2}h, w + \frac{1}{2}k_1)$

$k_3 = hf(t + \frac{1}{2}h, w + \frac{1}{2}k_2)$

$k_4 = hf(t + h, w + \frac{1}{2}k_2)$

$w = w + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$t = a + ih$

Βήμα 4 Εκτύπωση των (t, w)

Βήμα 5 Τέλος αλγορίθμου

<u>Αλγόριθμος Kutta</u>

Βήμα 1 Εισαγωγή των άκρων a, b του [a, b]

Εισαγωγή του πλήθους N

Προσδιορισμός της αρχικής συνθήκης

$w = w_0$

Βήμα 2 Υπολογισμός του βήματος h

$h = (b – a) / N$

Προσδιορισμός του πρώτου κομβικού σημείου

$t = a$

Βήμα 3 Καθώς i = 0, 1, 2, …, N – 1

$k_1 = hf(t, w)$

$k_2 = f(t + \frac{1}{3}h, w + \frac{1}{3}hk_1)$

$k_3 = f(t + \frac{2}{3}h, w - \frac{1}{3}hk_1 + hk_2)$

$k_4 = f(t + h, w + hk_1 - hk_2 + hk_3)$

$w=w+\frac{h}{8}(k_1+3k_2+3k_3+k_4)$

$t=t+h$

Βήμα 4 Εκτύπωση (t,w)

Βήμα 5 Τέλος αλγορίθμου

Αλγόριθμος Gill

Βήμα 1 Εισαγωγή των άκρων a,b του [a,b]

Εισαγωγή του πλήθους N

Προσδιορισμός της αρχικής συνθήκης

$w=w_0$

Βήμα 2 Υπολογισμός του βήματος h

$h=(b-a)/N$

Προσδιορισμός του πρώτου κομβικού σημείου

$t=a$

Βήμα 3 Καθώς i=0,1,…,N-1

$k_1=f(t,w)$

$k_2=f(t+\frac{1}{2}h,w+\frac{1}{2}hk_1)$

$k_3=f(t+\frac{1}{2}h,w+(-\frac{1}{2}+1/\sqrt{2})hk_1+(1-1/\sqrt{2})hk_2)$

$k_4=f(t+h,w-1/\sqrt{2}hk_2+(1+1/\sqrt{2})hk_3)$

$w=w+h/6[k_1+2(1-1/\sqrt{2})k_2+2(1+1/\sqrt{2})k_3+k_4]$

$t=t+h$

Βήμα 4 Εκτύπωση (t,w)

Βήμα 5 Τέλος αλγορίθμου

Παράδειγμα 2.5:

Να λυθεί η κανονική διαφορική εξίσωση

$y'=-y+t+1 \quad 0\le t\le 1 \ , \ y(0)=1$

με τις μεθόδους RungeKutta τέταρτης τάξης, Kutta και Gill

Λύση:

Στον πίνακα που ακολουθεί παρουσιάζονται τα αποτελέσματα της λύσης με τις τρεις μεθόδους. Οι προσεγγιστικές τιμές υπολογίστηκαν με τα προγράμματα RungeKutta4,KuttaMethod και GillMethod που δημιούργησε η συγγραφέας της διπλωματικής σε Fortran και παρουσιάζονται αναλυτικά στο παράρτημα .

κομβικά σημεία	**ακριβής λύση**	**RungeKutta4**	**KuttaMethod**	**Gill Method**
0	1	1	1	1
0,1	1,004837	1.004838	1.000498	1.000498
0,2	1,018731	1.018731	1.001987	1.001984
0,3	1,040818	1.040818	1.004455	1.004447
0,4	1,07032	1.070320	1.007894	1.007878 1.012267
0,5	1,106531	1.106531	1.012294	1.012267
0,6	1,148812	1.148812	1.017645	1.017604
0,7	1,196585	1.196586	1.023938	1.023881
0,8	1,249329	1.249329	1.031163	1.031088
0,9	1,30657	1.306570	1.039312	1.039215
1	1,367879	1.367880	1.048374	1.048254

■

2.3.5 Σύγκριση των μεθόδων RungeKutta και Taylor

Οι τύποι RungeKutta κάθε τάξης μοιάζουν με τα πολυώνυμα Taylor της αντίστοιχης τάξης. Μάλιστα για να υπολογιστούν οι συντελεστές των τύπων χρησιμοποιήθηκαν τα πολυώνυμα Taylor όπως είδαμε σε προηγούμενες παραγράφους. Οι τύπου RungeKutta πλεονεκτούν έναντι της μεθόδου Taylor καθώς δεν απαιτούν υπολογισμό παραγώγων ανώτερης τάξης της y(x).Επειδή οι διαφορικές εξισώσεις που συναντάμε στις διαφορικές εφαρμογές είναι συχνά πολύπλοκες, ο υπολογισμός των παραγώγων είναι πολύπλοκος. Οι τύποι RungeKutta απαιτούν υπολογισμό της f(x,y) στα κομβικά σημεία που είναι πιο εύκολο.

Τα μειονεκτήματα που παρουσιάζει η μέθοδος RungeKutta είναι η δυσκολία στον υπολογισμό των σφαλμάτων. Για την μελέτη των σφαλμάτων στην μέθοδο Taylor χρησιμοποιούμε το προηγούμενο βήμα της μεθόδου. Στην μέθοδο RungeKutta για την μελέτη των σφαλμάτων χρειάζεται να παρακολουθούνται οι τιμές των k. Αν διαφέρουν πολύ θα πρέπει να μειωθεί το βήμα h.

2.3.6 Μελέτη και έλεγχος του σφάλματος

Μία ιδανική μέθοδος διαφορικής εξίσωσης

$w_{i+1}=w_i+h_i\ \varphi(t_i,h_i,w_i)\ \ i=0,1,\ldots,N-1$

για την προσέγγιση της λύσης y(t) του προβλήματος αρχικών τιμών

$y'=f(t,y)\ \ a\le t\le b\ \ ,\ y(a)=a$

Θα έχει την ιδιότητα όταν ένας παράγοντας ανοχής ε>0 δίνεται, θα χρησιμοποιείται το ελάχιστο πλήθος κομβικών σημείων ώστε το συνολικό σφάλμα $|y(t_i)-w_i|$ δεν θα ξεπερνάει το ε για κάθε i=0,1,2,…N

Έχοντας έναν ελάχιστο αριθμό κομβικών σημείων και ελέγχοντας το συνολικό σφάλμα της διαφορικής εξίσωσης παρατηρούμε ότι συνάδουν με τα σημεία που είναι εξίσου ομοιόμορφα κατανεμημένες στο διάστημα [a,b].

Στην συνέχεια θα εξετασθεί μία τεχνική που μπορεί να χρησιμοποιηθεί για τον έλεγχο του σφάλματος της διαφορικής μεθόδου με αποτελεσματικό τρόπο επιλέγοντας κατάλληλα τα κομβικά σημεία.

Η μέθοδος του Euler

$w_0=a$

$w_{i+1}=w_i+hf(t_i,w_i)$

έχει τοπικό σφάλμα της τάξης O(h) που δίνεται από τον τύπο

$\tau_{i+1}=\frac{1}{h}\,[y(t_{i+1})-y(t_i)]-f(t_i,y(t_i))$

Η τροποποιημένη μέθοδος του Euler

$\widetilde{w}_{0=}\alpha$

$\widetilde{w}_{i+1}=w_i+h/2[f(t_i,w_i)+f(t_{i+1},w_i+hf(t_i,w_i)]$

έχει τοπικό σφάλμα τ_{i+1} της τάξης $O(h^2)$

Υποθέτοντας ότι

$w_i=y(t_i)=\widetilde{w}_i$ τότε

$y(t_{i+1})-w_{i+1}=y(t_{i+1})-w_i-hf(t_i,w_i)$

$\approx y(t_{i+1})-y(t_i)-hf(t_i,y(t_i))$

$=h\tau_{i+1}$

Επομένως

$$\tau_{i+1} \approx 1/h[y(t_{i+1})-w_{i+1}]$$

$$=1/h[y(t_{i+1})-\widetilde{w}_{i+1}]+1/h(\widetilde{w}_{i+1}-w_{i+1})$$

$$\approx\tilde{\tau}_{i+1}+1/h(\widetilde{w}_{i+1}-w_{i+1})$$

Όμως το τ_{i+1} είναι της τάξης O(h) ενώ το $\tilde{\tau}_{i+1}$ είναι της τάξης $O(h^2)$

επομένως $\tau_{i+1} \approx 1/h(\widetilde{w}_{i+1}-w_{i+1})$ μπορεί να χρησιμοποιηθεί για την προσέγγιση του τοπικού σφάλματος αποκοπής.

2.4 Οι μέθοδοι ανώτερης τάξης

2.4.1. Η μέθοδος RungeKutta –Fehlberg

Η μέθοδος RungeKutta –Fehlberg δημιουργήθηκε από τον Fehlberg και αποτελείται από μία μέθοδο Runge Kutta με τοπικό σφάλμα αποκοπής $O(h^2)$ την

$$\widetilde{w}_{i+1}=w_i+\frac{16}{135}k_1+\frac{6656}{12825}k_3+\frac{28561}{56430}k_4-\frac{9}{50}k_5+\frac{2}{55}k_6$$

για την εκτίμηση του τοπικού σφάλματος σε μία μέθοδο RungeKutte τέταρτης τάξης

$$w_{i+1}=w_i+\frac{25}{26}k_1+\frac{1408}{2565}k_3+\frac{2197}{4104}k_4-\frac{1}{5}k_5$$

όπου $k_1=hf(t_i,w_i)$

$$k_2=hf(t_i+\frac{1}{4}h,w_i+\frac{1}{4}k_1)$$

(2.28)

$$k_3=hf(t_i+\frac{3}{8}h,\ w_i+\frac{3}{32}k_1+\frac{9}{32}k_2)$$

$$k_4=hf(t_i+\frac{12}{13}h\ ,\ w_i+\frac{1932}{2197}k_1-\frac{7200}{2197}k_2+\frac{7296}{2197}k_3)$$

$$k_5=hf(t_i+h\ ,\ w_i+\frac{439}{216}k_1-8k_2+\frac{3680}{513}k_3-\frac{845}{4104}k_4)$$

$$k_6=hf(t_i+\frac{1}{2}h\ ,\ w_i-\frac{8}{27}k_1+2k_2-\frac{3544}{2565}k_3+\frac{1859}{4104}k_4-\frac{11}{40}k_5)$$

Το πλεονέκτημα της μεθόδου αυτής είναι ότι απαιτούνται μόνο έξι υπολογισμοί της f σε κάθε επανάληψη, ενώ οι μέθοδοι RungeKutta τέταρτης και πέμπτης τάξης εάν χρησιμοποιηθούν μαζί απαιτούν δέκα υπολογισμούς της f σε κάθε βήμα.

Στην θεωρία σφαλμάτων μία αρχική τιμή h στο i-βήμα χρησιμοποιείται για την εύρεση της πρώτης w_{i+1} και της $\tilde{w}_{i+1}$ το οποίο οδηγεί στον προσδιορισμό του q για αυτό το βήμα. Στην συνέχεια οι υπολογισμοί είναι επαναλαμβανόμενοι. Η διαδικασία αυτή απαιτεί το διπλάσιο σε αριθμό εκτιμήσεως από ότι χωρίς τον έλεγχο λάθους. Στην πράξη η τιμή του q για να χρησιμοποιηθεί επιλέγεται κάπως διαφορετικά για να αυξηθεί το κόστος. Η τιμή του q που προσδιορίζεται στο i-βήμα χρησιμοποιείται για δύο λόγους:

1.Για να απορρίπτει την αρχική επιλογή του h στο i-βήμα εάν είναι απαραίτητο

και επαναλαμβάνεται η διαδικασία χρησιμοποιώντας το qh.

2.Για να «προβλέπει» την κατάλληλη αρχική επιλογή του h για το (i+1)-βήμα.

Εξαιτίας της «ποινής» που πρέπει να αποδίδεται όταν πολλά βήματα επαναλαμβάνονται το q τείνει να επιλέγεται συντηρητικά. Στην πραγματικότητα για την μέθοδο RungeKutta-Fehlberg με n=4 η συνήθης επιλογή είναι

$$q=\left(\frac{\varepsilon h}{2|\tilde{w}_{i+1}-w_i|}\right)^{1/4}=0.84\left(\frac{\varepsilon h}{2|\tilde{w}_{i+1}-w_i|}\right)^{1/4}$$

Στον αλγόριθμο που ακολουθεί χρησιμοποιείται η μέθοδος RungeKutta-Fehlberg με έλεγχο σφάλματος

<u>Αλγόριθμος RungeKutta – Fehlberg</u>

Βήμα 1 Εισαγωγή των άκρων a, b του [a, b]

Εισαγωγή του πλήθους N

Προσδιορισμός της tolerance TOL

Εισαγωγή του μέγιστου βήματος hmax και του ελάχιστου βήματος hmin

Βήμα 2 Προσδιορισμός της αρχικής συνθήκης

$w = w_0$

Υπολογισμός h=hmax και t=α

Βήμα 3 Καθώς t≤b τότε εκτέλεσε τα Β.4 – Β.11

Βήμα 4 $k_1=hf(t_i,w_i)$

$k_2=hf(t_i+\frac{1}{4}h,w_i+\frac{1}{4}k_1)$

$k_3=hf(t_i+\frac{3}{8}h,\ w_i+\frac{3}{32}k_1+\frac{9}{32}k_2)$

$k_4=hf(t_i+\frac{12}{13}h\ ,\ w_i+\frac{1932}{2197}k_1-\frac{7200}{2197}k_2+\frac{7296}{2197}k_3)$

$k_5=hf(t_i+h\ ,\ w_i+\frac{439}{216}k_1-8k_2+\frac{3680}{513}k_3-\frac{845}{4104}k_4)$

$k_6=hf(t_i+\frac{1}{2}h\ ,\ w_i-\frac{8}{27}k_1+2k_2-\frac{3544}{2565}k_3+\frac{1859}{4104}k_4-\frac{11}{40}k_5)$

Βήμα 5 $R=\frac{1}{h}\left|\frac{1}{360}k_1-\frac{128}{4275}k_3-\frac{2197}{75240}k_4+\frac{1}{50}k_5+\frac{2}{55}k_6\right|$

Βήμα 6 $\delta=0.84(TOL/R)^{1/4}$

Βήμα 7 Εάν $R\leq TOL$ τότε εκτέλεσε το βήμα 8

Βήμα 8 Θέτουμε $t=t+h$

$w_{i+1}=w_i+\frac{25}{26}k_1+\frac{1408}{2565}k_3+\frac{2197}{4104}k_4-\frac{1}{5}k_5$

Βήμα 9 Εάν $\delta\leq 0.1$ τότε υπολογισμός $h=0.1h$

Διαφορετικά εάν $\delta\geq 4$ τότε

υπολογισμός $h=\delta h$

Βήμα 10 Εάν $h>hmax$ τότε

$h=hmax$

Βήμα 11 Εάν $h<hmin$ τότε

εκτύπωσε ελάχιστο h επετεύχθη

Βήμα 12 Τέλος αλγορίθμου

Παράδειγμα 2.6:

Να εφαρμοστεί η μέθοδος RungeKutta -Fehlberg στο πρόβλημα αρχικών τιμών:

$y' = -y + t + 1, \quad 0 \le t \le 1, \quad y(0) = 1.$

Λύση:

Στον πίνακα παρουσιάζονται οι ακριβείς και οι προσεγγιστικές λύσεις σε κάθε κομβικό σημείο. Οι προσεγγιστικές τιμές είναι τα αποτελέσματα του προγράμματος RungeKutta -Fehlberg που δημιούργησε η συγγραφέας της διπλωματικής σε Fortran. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα.

κομβικά σημεία	**ακριβης λύση**	**προσεγγιστική λύση**	**R**	**h**	**d**
0	1	1	$1{,}3087417*10^{-7}$	0,1	0
0,1	1,004837	1,004837	$1{,}23013088*10^{-7}$	0,1	3.713712
0,2	1,018731	1,018731	$1{,}0627485*10^{-7}$	0,1	3.771666
0,3	1,040818	1,040818	$1{,}0275755*10^{-7}$	0,1	3.912138
0,4	1,07032	1,070320	$8{,}8467623*10^{-8}$	0,1	3.945194
0,5	1,106531	1,106531	$8{,}1265398*10^{-8}$	0,1	4.095677
0,6	1,148812	1,148812	$8{,}4158727*10^{-8}$	0,1	4.183554
0,7	1,196585	1,196585	$7{,}4158727*10^{-8}$	0,1	4.280369
0,8	1,249329	1,249329	$6{,}5205256*10^{-8}$	0,1	4.420294
0,9	1,30657	1,306570	$5{,}9072487*10^{-8}$	0,1	4.530806

1	1,367879	1,367880	$5{,}2529959*10^{-8}$	0,1	4.665734

■

2.4.2 Η μέθοδος Fehlberg 5ης τάξης

Μέθοδοι ανώτεροι της 4ης τάξης αναφέρονται αρκετές στην βιβλιογραφία, αλλά η μέθοδος που είναι πιο διαδεδομένη είναι η μέθοδος Fehlberg 5ης τάξης. Στην μέθοδο αυτή χρησιμοποιούνται έξη βοηθητικές παράμετροι K για την περιγραφή της. Συγκεκριμένα:

$k_1 = hf(x_i, w_i)$

$k_2 = hf\left(x_i + \frac{h}{6}, w_i + \frac{k1}{6},\right)$

$k_3 = hf\left(x_i + \frac{4h}{6}, w_i + \frac{4k1}{75} + \frac{16\,k2}{75}\right)$ (2.30)

$k_4 = hf\left(x_i + \frac{2h}{3}, w_i + \frac{5\,k1}{6} - \frac{8\,k2}{3} + \frac{5\,k3}{2}\right)$

$k_5 = h\,f\left(x_i + \frac{4h}{5}, w_i - \frac{8\,k1}{5} + \frac{144\,k2}{25} - 4k_3 + \frac{16\,k4}{25}\right)$

$k_6 = h\,f\left(x_i, w_i + \frac{361\,k1}{320} - \frac{18\,k2}{5} + \frac{407\,k3}{128} - \frac{11\,k4}{80} + \frac{55\,k5}{128}\right)$

$w_{i+1} = w_i + \frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4)$ για i=0,1,…,N

Παράδειγμα 2.7

Να εφαρμοστεί η μέθοδος Fehlberg 5ης τάξης στο πρόβλημα αρχικών τιμών

$y' = -y + t + 1, \quad 0 \le t \le 1, \quad y(0) = 1.$

Λύση:

Στον πίνακα παρουσιάζονται οι ακριβείς και οι προσεγγιστικές λύσεις σε κάθε κομβικό σημείο, καθώς και τα σφάλματα αποκοπής. Οι προσεγγιστικές τιμές είναι τα αποτελέσματα του προγράμματος Fehlberg5 που δημιούργησε η συγγραφέας της διπλωματικής σε Fortran. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα.

κομβικά σημεία	**ακριβής λύση**	**προσεγγιστική λύση**	**Fehlberg5 Error**
0	1	1	0
0,1	1,004837	1,003806	0,001031
0,2	1,018731	1,017067	0,001664
0,3	1,040818	1,038856	0,001962
0,4	1,07032	1,068332	0,001988
0,5	1,106531	1,104742	0,001789
0,6	1,148812	1,147402	0,00141
0,7	1,196585	1,195699	0,000886
0,8	1,249329	1,249079	0,00025
0,9	1,30657	1,307042	0,000472
1	1,367879	1,369137	0,001258

2.5 Εφαρμογές

Εφαρμογή 2.1:

Στη μη αναστρέψιμη χημική αντίδραση στην οποία δύο μόρια υγρού διχρωμικού καλίου ($K_2Cr_2O_7$) ,δύο μόρια νερού (H_2O) και τρία άτομα υγρού θείου (S) συνδυάζονται και παράγουν τρία μόρια διοξειδίου του θείου (SO_2) σε αέρια μορφή, τέσσερα μόρια υδροξείδιο του καλίου σε υγρή μορφή (KOH) και δύο μόρια υγρού χρωμικού οξέος (Cr_2O_3) η οποία μπορεί να αναπαρασταθεί γραφικά από την μορφή

$$2K_2Cr_2O_7 + 2H_2O + 3S \rightarrow 4KOH + 2Cr_2O_3 + 3SO_2$$

Εάν n_1 μόρια $K_2Cr_2O_7$,n_2 μόρια νερού και n_3 μόρια θείου είναι αρχικά διαθέσιμα, η επόμενη διαφορική εξίσωση περιγράφει το ποσό x(t) του KOH που παράγεται μετά από χρόνο t.

$\frac{dx}{dt}=k(n_1-\frac{x}{2})^2(n_2-\frac{x}{2})^2(n_3-\frac{3x}{4})^3$

Όπου k είναι η ταχύτητα της αντίδρασης. Εάν $k=6.22*10^{-19}$, $n_1 = n_2 = 1000$ και $n_3 =1500$ πόσα μόρια υδροξείδιο του θείου θα έχουν σχηματιστεί μετά από 2 δευτερόλεπτα ; Χρησιμοποιήστε την μέθοδο Runge- Kutta 4ης τάξης με βήμα h=0.1

Λύση:

Στον πίνακα που ακολουθεί εμφανίζονται οι προσεγγιστικές λύσεις του προβλήματος όπως προέκυψαν από το πρόγραμμα IrreversibleChemicalReactionRK4 σε Fortran που δημιούργησε η συγγραφέας της διπλωματικής. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα.

κομβικά σημεία ti	**προσεγγίσεις**
0	0
0,1	$2,0992*10^{-34}$
0,2	$4,1984*10^{-34}$
0,3	$6,2977*10^{-34}$
0,4	$8,3969*10^{-34}$
0,5	$1,0496*10^{-33}$
0,6	$1,2595*10^{-33}$
0,7	$1,4694*10^{-33}$
0,8	$1,6794*10^{-33}$
0,9	$1,8893*10^{-33}$
1	$2,0992*10^{-33}$
1,1	$2,3091*10^{-33}$
1,2	$2,5191*10^{-33}$
1,3	$2,7290*10^{-33}$
1,4	$2,9389*10^{-33}$
1,5	$3,1488*10^{-33}$
1,6	$3,3588*10^{-33}$
1,7	$3,5687*10^{-33}$
1,8	$3,7786*10^{-33}$
1,9	$3,9885*10^{-33}$
2	$4,1985*10^{-33}$

Εφαρμογή 2.2:

Οι διαφορικές εξισώσεις χρησιμοποιούνται ευρύτατα για την μοντελοποίηση πολλών διαφορικών φυσικών φαινομένων και συστημάτων και κυρίως αυτών που εξελίσσονται στο χρόνο. Ένα τέτοιο μοντέλο, το οποίο περιγράφεται από μια διαφορική εξίσωση, έχει αναπτυχθεί από επιστήμονες με στόχο τη μελέτη του πληθυσμού των σκουληκιών Spruce Budworm σε σχέση με τον χρόνο και άλλων παραγώγων

Το έντομο Spruce Budworm (με επιστημονική ονομασία Choristoneura fumiferana-Clemens) είναι ένα από τα καταστρεπτικότερα έντομα για τα δάση ελάτων και λοιπόν κωνοφόρων δέντρων στις περιοχές του Καναδά. Το έντομο έχει υψηλή αναπαραγωγική ικανότητα, αλλά ο πληθυσμός του συγκρατείται λόγω φυσικών παραγόντων. Παρόλα αυτά κάτω από ευνοϊκές συνθήκες υπάρχουν ξεσπάσματα, τα οποία είναι καταστρεπτικά για ολόκληρα δάση. Εκτιμάται από την Δασική Υπηρεσία του Καναδά ότι το καταστροφικό φαινόμενο εμφανίζεται με περιοδικότητα, με περίοδο τα 29 χρόνια.

Το μαθηματικό μοντέλο που αναπτύχθηκε από επιστήμονες (R. Morris, D. Ludwing, D. Jones, C.S. Holling) στο πανεπιστήμιο British Columbia είναι το ακόλουθο:

$$\frac{dx}{dt} = rx\left(1-\frac{x}{K}\right) - \left(bpx^2\right)\left(x^2+a^2\right)^{-1}$$

Όπου

x: πληθυσμός των σκουληκιών

r: φυσικός ρυθμός ανάπτυξης σκουληκιών

K: επίπεδο κορεσμού

b: μέτρο αρπακτικότητας (ικανότητας των πουλιών να πιάνουν σκουλήκια)

p: επίπεδο πληθυσμού πουλιών

α: παράμετρος που καθορίζει το επίπεδο του πληθυσμού των σκουληκιών ώστε να προσελκύονται τα πουλιά

Επίσης έχει παρατηρηθεί ότι οι παράμετροι α και K εξαρτώνται άμεσα από την επιφάνεια του φυλλώματος S κάθε δένδρου και κατά προσέγγιση είναι:

$$\left.\begin{array}{l}\alpha=0.5S\\ K=4S\end{array}\right\} \Rightarrow \quad K=8a$$

Στον πίνακα που ακολουθεί εμφανίζονται τα αποτελέσματα που προκύπτουν από την επίλυση της παραπάνω διαφορικής εξίσωσης με την μέθοδο Runge Kutta με το πρόγραμμα SpruceBudworm που δημιουργήθηκε από την συγγραφέα της αυτής της διπλωματικής σε Fortran. Το πρόγραμμα παρουσιάζεται αναλυτικά στο παράρτημα. Οι τιμές των παραμέτρων είναι:

r=0.8

K=43

b=1.4

p=1.6

a=1.2

x_0=5

στο διάστημα [A,B]=[0,20]

κομβικά σημεία ti	**προσεγγίσεις**
0	5
1	6,896798
2	10,07586
3	14,85877
4	20,92000
5	27,10389
6	32,15837
7	35,59227
8	36,64220
9	38,77277
10	39,36919
11	39,67646
12	39,83282
13	39,31190
14	39,95176
15	39,97182
16	39,98191
17	39,98699
18	39,98954
19	39,99082
20	39,99155

ΚΕΦΑΛΑΙΟ 3^{ο} ΠΟΛΥΒΗΜΑΤΙΚΕΣ ΜΕΘΟΔΟΙ

3.1 Εισαγωγή

Οι μέθοδοι που μελετήθηκαν στα προηγούμενα κεφάλαια ονομάζονται μέθοδοι απλού βήματος (one- step methods) καθώς για τον υπολογισμό της προσέγγιση στο κομβικό σημείο t_{i+1} χρησιμοποιούνται μόνο πληροφορίες από το προηγούμενο κομβικό σημείο t_i. Αν και οι μέθοδοι απλού βήματος υπολογίζουν τιμές της συνάρτησης στο διάστημα $[t_i , t_{i+1}]$, δεν διατηρούν αυτή την πληροφορία για άμεση μελλοντική χρήση στις επόμενες προσεγγίσεις. Όλες οι πληροφορίες που χρησιμοποιούνται στις μεθόδου απλού βήματος κατά συνέπεια υπολογίζονται μέσα στο διάστημα στο οποίο προσεγγίζουν την λύση.

Καθώς η προσέγγισης της λύσης είναι διαθέσιμη σε κάθε ένα κομβικό σημείο t_0, $t_1,t_2,\ldots,t_i$ πριν από την προσέγγιση στο σημείο t_{i+1} , και καθώς το σφάλμα $|w_i - y(t_i)|$ αυξάνεται όσο αυξάνεται το i, είναι λογικό να αναπτυχθούν μέθοδοι που να χρησιμοποιούν τα προηγούμενα ακριβή δεδομένα , όταν προσεγγίζουν την λύση στο κομβικό σημείο t_{i+1}.

3.2 Μέθοδοι Adams – Bashforth 4^{ης} τάξης

Ορισμός 4.1

Πολυβηματική μέθοδος για την επίλυση του προβλήματος αρχικών τιμών

$y' = f(t,y)$ $a \le t \le b$ και $y(a)=a$

ονομάζεται η μέθοδος στην οποία η διαφορική εξίσωση που χρησιμοποιείται για την προσέγγιση w_{i+1} στο κομβικό σημείο t_{i+1} μπορεί να δοθεί από την εξίσωση:

$$w_{i+1} = a_{m-1}w_i + a_{m-2}w_{i-1} + \ldots + a_0 w_{i+1-m} +$$

$$+h\,[\,b_m f(t_{i+1}, w_{i+1}) + b_{m-1} f(t_i, w_i) + \ldots + b_0 f(t_{i+1-m}, w_{i+1-m})] \qquad (3.1)$$

για $i=m-1,m-2,\ldots,N-1$ όπου m ακέραιος με $m>1$ και οι αρχικές συνθήκες

$w_0=a_0$

$w_1= a_1$

$w_2=a_2$

…

$w_{m-1}= a_{m-1}$

προσδιορίζονται και τα βήμα h υπολογίζεται από την σχέση h=(b-a)/N .

Όταν $b_m=0$ οι μέθοδοι ονομάζονται άμεσες (explicit ή open methods) και υπολογίζουν την προσέγγιση w_{i+1} άμεσα χρησιμοποιώντας τις τιμές που είχαν προσδιοριστεί προηγούμενα.

Όταν $b_m \neq 0$, οι μέθοδοι ονομάζονται άμεσες (implicit ή closed methods) όπου η προσέγγιση w_{i+1} εμφανίζεται και στις δύο πλευρές της σχέσης (3.1) και προσδιορίζονται με την βοήθεια κάποιου έμμεσου τρόπου.

Η μέθοδος Adams– Bashforth $4^{ης}$ τάξης

Η εξίσωση που ακολουθεί δίνει την άμεση μέθοδο, όπου χρειάζονται 4 τιμές της w_i για την επίλυσή της

$w_0=a_0$,$w_1= a$,$w_2=a_2$, $w_3=a_3$

$$w_{i+1} = w_i + \frac{h}{24} [\, 55\, f(t_i, w_i) – 59\, f(t_{i-1}, w_{i-1}) + 37\, f(t_{i-2}, w_{i-2}) – 9\, f(t_{i-3}, w_{i-3})] \qquad (3.2)$$

i=3,4,…,N-1

Η μέθοδος Adams- Moulton $4^{ης}$ τάξης

$w_0=a_0$, $w_1= a$, $w_2=a_2$

$$w_{i+1}= w_i + \frac{h}{24} [\, 9\, f(t_{i-1} , w_{i-1}) + 19\, f(t_i, w_i) – 5\, f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})] \qquad (3.3)$$

Η εξίσωση (3.3) ορίζει μία έμμεση μέθοδο τριών βημάτων γνωστή ως Adams-Moulton $4^{ης}$ τάξης.

Παρατήρηση 3.1:

Οι αρχικές τιμές στις εξισώσεις (3.2) και (3.3) προσδιορίζονται στην γενική περίπτωση θεωρώντας $a_0=a$

και υπολογίζοντας τις υπόλοιπες τιμές από μία μέθοδο απλού βήματος. Συνήθως χρησιμοποιείται η μέθοδος Runge Kutta $4^{ης}$ τάξης.

Ορισμός 3.2

Έστω y(t) η λύση στο πρόβλημα αρχικών τιμών

$$y' = f(t,y) \quad a \leq t \leq b \text{ και } y(a)=a$$

$$w_{i+1} = a_{m-1} w_i + a_{m-2}\, w_{i-1} + \ldots + a_0\, w_{i+1-m} +$$

$$+h [\, b_{m+1}\, f(\, t_{i+1} , w_{i+1}) + b_{m-1}\, f(\, t_i, w_i) + \ldots + b_0\, f(\, t_{i+1-m}, w_{i+1-m})]$$

είναι το (i+1)- βήμα πολυβηματικής μεθόδου, τότε το τοπικό σφάλμα απκοπής σε αυτό το βήμα είναι τ_{i+1} και υπολογίζεται από τον τύπο

$$\tau_{i+1} = \frac{1}{h}[y(t_{i+1}) - a_m y(t_i) - a_0 y(t_{i-m})] - [b_{m+1} f(t_{i+1}, y(t_{i+1})) + \ldots + b_0 f(t_{i-m}, y(t_{i-m}))] \quad (3.4)$$

για i=m,m+1,…, N-1 .

3.3 Άμεσες μέθοδοι

Κάποιες άμεσες πολυβηματικές μέθοδοι παρουσιάζονται στη συνέχεια μαζί με τις απαιτούμενες αρχικές τιμές και το τοπικό σφάλμα αποκοπής.

Μέθοδος Adams – Bashforth 3- βημάτων

$w_0=a_0$, $w_1= a$, $w_2=a_2$

$$w_{i+1} = w_i + \frac{h}{12}[23 f(t_i, w_i) - 16 f(t_{i-1}, w_{i-1}) + 5 f(t_{i-2}, w_{i-2})] \quad (3.5)$$

όπου i=2,3,…,N-1,

$$\tau_{i+1} = \frac{3}{8} y^{(4)}(\mu_i) h^3 \quad (3.6)$$

Μέθοδος Adams – Bashforth 4- βημάτων

$w_0=a_0$,$w_1= a$,$w_2=a_2$, $w_3=a_3$

$$w_{i+1} = w_i + \frac{h}{24}[55 f(t_i, w_i) - 59 f(t_{i-1}, w_{i-1}) + 37 f(t_{i-2}, w_{i-2}) - 9 f(t_{i-3}, w_{i-3})] \quad (3.7)$$

i=3,4,…,N-1

$$\tau_{i+1} = \frac{251}{720} y^{(5)}(\mu_i) h^4 \quad (3.8)$$

Μέθοδος Adams – Bashforth 5- βημάτων

$w_0=a_0$, $w_1= a_1$, $w_2=a_2$, $w_3=a_3$, $w_4=a_4$

$$w_{i+1} = w_i + \frac{h}{720}[1901 f(t_i, w_i) - 2774 f(t_{i-1}, w_{i-1}) + 2616 f(t_{i-2}, w_{i-2}) - 1271 f(t_{i-3}, w_{i-3}) + 251 f(t_{i-4}, w_{i-4})] \quad (3.9)$$

i=4,…,N-1

$$\tau_{i+1} = \frac{95}{288} y^{(6)}(\mu_i) h^5 \quad (3.10)$$

3.4 Έμμεσες μέθοδοι

Κάποιες από τις πιο συνηθισμένες έμμεσες μεθόδους παρουσιάζονται στην συνέχεια.

Adams-Moulton 2-βημάτων

$w_0=a_0$, $w_1= a$

$$w_{i+1}= w_i + \frac{h}{12} [5 f(t_{i+1} , w_{i+1}) + 8 f(t_i , w_i) - f(t_{i-1}, w_{i-1})] \quad (3.11)$$

για i=1,2,…,N-1

$$\tau_{i+1}=- \frac{1}{24} y^{(4)}(\mu_i)h^3 \quad (3.12)$$

Adams-Moulton 3-βημάτων

$w_0=a_0$, $w_1= a$, $w_2=a_2$

$$w_{i+1}= w_i + \frac{h}{24} [9 f(t_{i+1} , w_{i+1}) + 19 f(t_i , w_i) - 5 f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})] \quad (3.13)$$

i=3,4,…,N-1

$$\tau_{i+1}=- \frac{3}{160} y^{(6)}(\mu_i)h^5 \quad (3.14)$$

Adams-Moulton 4-βημάτων

$w_0=a_0$,$w_1= a$,$w_2=a_2$, $w_3=a_3$

$$w_{i+1} = w_i + \frac{h}{720} [251 f(t_{i+1} , w_{i+1}) + 646 f(t_i, w_i) – 264 f(t_{i-1}, w_{i-1}) + 106 f(t_{i-2}, w_{i-2}) – 19f(t_{i-3}, w_{i-3})]$$

i=3,4,…,N-1 (3.15)

$$\tau_{i+1}=- \frac{3}{160} y^{(6)}(\mu_i)h^5$$

3.5 Παρατηρήσεις και εφαρμογές άμεσων και έμμεσων μεθόδων

Ενδιαφέρον παρουσιάζει η σύγκριση μιας m- βημάτων Adams- Bashforth άμεσης μεθόδου με μία (m-1)- βημάτων έμμεσης μεθόδου Adams- Moulton. Αμφότερες οι μέθοδοι απαιτούν m- υπολογισμούς της f σε κάθε βήμα και έχουν τοπικό σφάλμα αποκοπής $O(h^m)$.

Γενικά, οι συντελεστές των όρων που περιέχουν την f και του τοπικού σφάλματος αποκοπής είναι μικρότεροι στις μεθόδους Adams-Moulton. Αυτό συνεπάγεται μεγαλύτερη ευστάθεια και μικρότερο σφάλμα για τις έμμεσες μεθόδους. Το γεγονός αυτό γίνεται εμφανές στο επόμενο παράδειγμα.

Παράδειγμα 3.1

Να εφαρμοστεί η μέθοδος Adams-Bashforth 4-βημάτων και Adams- Moulton 3-βημάτων στο πρόβλημα αρχικών τιμών

$$y' = -y + t + 1, \qquad 0 \le t \le 1, \qquad y(0) = 1$$

χρησιμοποιώντας h=0.1 και τις απαιτούμενες αρχικές τιμές από την ακριβή λύση της διαφορικής εξίσωσης.

Λύση:

Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο η ακριβής λύση της διαφορικής εξίσωσης είναι

$y(t)= e^{-t} + t.$

Στον πίνακα που ακολουθεί παρουσιάζονται οι προσεγγιστικές τιμές w_i με τις παραπάνω μεθόδους χρησιμοποιώντας τα προγράμματα AdamsBashforth4 και AdamsMoulton3 που δημιουργήθηκαν από την συγγραφέα της διπλωματικής σε Fortran. Στο παράρτημα παρουσιάζονται αναλυτικά τα παραπάνω προγράμματα.

κομβικά σημεία	**ακριβής λύση**	**Adams Bashforth4**	**Adams Moulton3**	**Adams Bashforth4 Error**	**Adams Moulton3 Error**
0	1	**1**	**1**	0	0
0,1	1,004837	**1.004837**	**1.004837**	0	0
0,2	1,018731	**1.018731**	**1.018731**	0	0
0,3	1,040818	**1.040818**	**1.040818**	0	0
0,4	1,07032	1.070323	1.062905	0.000003	0.007415
0,5	1,106531	1.106535	1.092685	0.000004	0.01385
0,6	1,148812	1.148818	1.130002	0.000006	0.01881
0,7	1,196585	1.196593	1.173930	0.000008	0.022655
0,8	1,249329	1.249338	1.223784	0.000009	0.025545
0,9	1,30657	1.306580	1.278945	0.00001	0.027625
1	1,367879	1.367890	1.338850	0.000011	0.029029

3.6 Μέθοδοι πρόβλεψης- διόρθωσης.

Στην πράξη, οι έμμεσες πολυβηματικές μέθοδοι δεν χρησιμοποιούνται για την εύρεση προσεγγιστικών λύσεων όπως παρουσιάστηκε παραπάνω. Κυρίως χρησιμοποιούνται για να βελτιώσουν τις προσεγγιστικές λύσεις των άμεσων μεθόδων. Ένας τέτοιος συνδυασμός μίας άμεσης και μίας έμμεσης μεθόδου ονομάζεται μέθοδος πρόβλεψης-διόρθωσης. Οι άμεσες μέθοδοι «προβλέπουν» την προσεγγιστική λύση και οι έμμεσες λύσεις χρησιμοποιώντας τις «προβλέψεις» αυτές τις βελτιώνουν προσεγγίζοντας αποτελεσματικότερα την ακριβή λύση.

Σε μια μέθοδο $4^{ης}$ τάξης, για την επίλυση ενός προβλήματος αρχικών τιμών, απαιτούνται στο 1^0 – βήμα οι αρχικές τιμές w_0, w_1, w_2 και w_3 . Οι τιμές αυτές συνήθως υπολογίζονται από την μέθοδο Runge- Kutta $4^{ης}$ – τάξης. Με την μέθοδο Adams – Bashforth με 4- βήματα υπολογίζεται η προσέγγιση $w^{(0)}_4$ της $y(t_4)$ χρησιμοποιώντας τον τύπο

$$w^{(0)}_4 = w_3 + \frac{h}{24} [55\, f(t_3 , w_3) - 52\, f(t_2 , w_2) + 37\, f(t_1, w_1) - 9\, f(t_0, w_0)] \qquad (3.16)$$

Στην συνέχεια η προσέγγιση αυτή βελτιώνεται από την μέθοδο τριών βημάτων Adams- Moulton χρησιμοποιώντας τον τύπο

$$w^{(1)}_4 = w_3 + \frac{h}{24} [9\, f(t_4 , w_4) + 19\, f(t_3 , w_3) - 5f(t_2 , w_2) + f(t_1, w_1)]$$

Η τελική τιμή $w^{(1)}_4$ είναι η τελική προσέγγιση της $y(t_4)$. Η διαδικασία αυτή συνεχίζεται για να βρεθούν οι $w^{(0)}_i$ και οι $w^{(1)}_4$ για i=4,5,…,N.

Αλγόριθμος Adams4PredictorCorrector

Βήμα 1 Εισαγωγή των άκρων a, b του [a, b]

Εισαγωγή του πλήθους N

Προσδιορισμός της αρχικής συνθήκης

$w = w_0$

Βήμα 2 Υπολογισμός του βήματος h

$h = (b – a) / N$

Προσδιορισμός του πρώτου κομβικού σημείου

$t = a$

Βήμα 3 Για i = 1,2,3 εκτέλεσε τα Β.4 – Β.5

Βήμα 4 υπολόγισε τα εξής

$k_1 = hf (t, w)$

$k_2 = hf\ (t + h/2,\ w + k_1/2)$

$k_3 = hf\ (t + h/2,\ w + k_2/2)$

$k_4 = hf\ (t + h,\ w + k_2/2)$

Βήμα 5 υπολόγισε τα εξής:

$w = w + (k_1 + 2k_2 + 2k_3 + k_4)/6$

$t = a + ih$

Βήμα 6 Για i= 4,5,…,N εκτέλεσε τα Β.7-Β.10

Βήμα 7 t=a+Ih

$w= w_3 + \frac{h}{24} [55\ f(t_3 , w_3) - 52\ f(t_2 , w_2) +37\ f(t_1, w_1) - 9\ f(t_0, w_0)]$

(τιμή πρόβλεψης)

$w= w_3 + \frac{h}{24} [9\ f(t_4 , w_4) + 19\ f(t_3 , w_3) - 5f(t_2 , w_2) +f(t_1, w_1)]$

(τιμή διόρθωσης)

Βήμα 8 Εκτύπωσε (t,w)

Βήμα 9 Για i=0,1,2

$t_i=t_{i+1}$

$w_i=w_{i+1}$

Βήμα 10 Εκχώρησε

$t_3=t$

$w_3=w$

Βήμα 11 Τέλος

Παράδειγμα 3.2

Να εφαρμοστεί η μέθοδος Adams – Bashforth 4- βημάτων στο πρόβλημα αρχικών τιμών

$$y' = -y + t + 1, \qquad 0 \le t \le 1, \qquad y(0) = 1$$

χρησιμοποιώντας h=0.1 και τις απαιτούμενες αρχικές τιμές υπολογίζονται από την μέθοδο Runge- Kutta $4^{ης}$.

Λύση:

Στον πίνακα που ακολουθεί παρουσιάζονται οι προσεγγιστικές τιμές w_i με τις παραπάνω μεθόδους χρησιμοποιώντας το πρόγραμμα AdamsFourthOrderPredictorCorrector που δημιουργήθηκαν από την συγγραφέα της διπλωματικής σε Fortran. Στο παράρτημα παρουσιάζεται αναλυτικά το παραπάνω πρόγραμμα.

κομβικά σημεία	**ακριβής λύση**	**Πρόβλεψη w_p**	**Διόρθωση W_c**	**Adams4OrderP-C Error**
0	1	Αρχική τιμή	Αρχική τιμή	0
0,1	1,004837	Αρχική τιμή	Αρχική τιμή	0
0,2	1,018731	Αρχική τιμή	Αρχική τιμή	0
0,3	1,040818	Αρχική τιμή	Αρχική τιμή	0
0,4	1,07032	1,070307	1,070303	0,000017
0,5	1,106531	1,110269	1,106374	0,0000157
0,6	1,148812	1,156191	1,148391	0,0000421
0,7	1,196585	1,207476	1,195786	0,000799
0,8	1,249329	1,263635	1,248048	0,001281
0,9	1,30657	1,324197	1,304714	0,001856
1	1,367879	1,388743	1,365365	0,002514

■

3.7 Σύγκριση των μεθόδων

Στην παράγραφο αυτή θα μελετήσουμε την αποτελεσματικότητα των μεθόδων που παρουσιάστηκαν σε αυτή την διπλωματική εργασία. Οι μέθοδοι ανάλογα με την τάξη τους και τα βήματά τους έχουν την δυνατότητα να προσεγγίζουν την ακριβή λύση στα κομβικά σημεία με μέτριο ή καλό τρόπο. Όσο αυξάνεται η τάξη των μεθόδων τόσο καλύτερες προσεγγίσεις παίρνουμε. Επίσης οι πολυβηματικές μέθοδοι και οι μέθοδοι πρόβλεψης διόρθωση παρουσιάζουν πάρα πολύ καλές προσεγγίσεις, όμως είναι αρκετά πιο πολύπλοκες. Ένα μέγεθος που εμφανίζει την αποδοτικότητα των μεθόδων είναι το τοπικό σφάλμα αποκοπής σε κάθε κομβικό σημείο. Στους παρακάτω πίνακες φαίνονται παραστατικά μόνο τα σφάλματα αποκοπής.

κομβικά σημεία	**ακριβής λύση**	**Σφάλμα Euler**	**Taylor2 Error**	**Taylor4Error**
0	1	0,00	0	0
0,1	1,004837	0,004837	0,000163	0
0,2	1,018731	0,008731	0,000294	0,000083
0,3	1,040818	0,011818	0,000400	0,000113

0,4	1,07032	0,014220	0,000482	0,000136
0,5	1,106531	0,016041	0,000545	0,000149
0,6	1,148812	0,017371	0,000591	0,000169
0,7	1,196585	0,018288	0,000625	0,000177
0,8	1,249329	0,018862	0,000646	0,000182
0,9	1,30657	0,019150	0,000657	0,000185
1	1,367879	0,019201	0,000662	0,000187

Όπως φαίνεται και από τον πίνακα η μέθοδος του Euler έχει μεγαλύτερο σφάλμα, από την μέθοδο του Taylor τάξης δύο και αυτή με την σειρά της μεγαλύτερο από την μέθοδο Taylor τέταρτης τάξης. Η πιο ακριβή απ΄αυτές τις τρεις μεθόδους είναι η μέθοδος Taylor $4^{ης}$ τάξης.

κομβικά σημεία	**ακριβής λύση**	**Modified Euler Error**	**Heun Method Error**	**Midpoint Method Error**	**RungeKutta 3 Error**
0	1	0	0	0	0
0,1	1,004837	0.000163	0.005163	0.000163	0.000004
0,2	1,018731	0.000294	0.000294	0.000294	0.000008
0,3	1,040818	0.00040	0.015282	0.0004	0.00001
0,4	1,07032	0.000482	0.02058	0.000482	0.000012
0,5	1,106531	0.000545	0.02491	0.000545	0.000014
0,6	1,148812	0.000591	0.029485	0.000591	0.000015
0,7	1,196585	0.000625	0.033882	0.000625	0.000015
0,8	1,249329	0.000646	0.038092	0.000646	0.000016
0,9	1,30657	0.00113	0.042108	0.000657	0.000017
1	1,367879	0.000662	0.045932	0.000662	0.000016

Από τις μεθόδους της οικογένειας Runge Kutta 2^{ης} τάξης, η μέθοδος Heun παρουσιάζει τα μεγαλύτερα σφάλματα ενώ οι μέθοδοι Modified Euler και Midpoint έχουν σχεδόν τα ίδια σφάλματα αποκοπής, όπου η μέθοδος Midpoint είναι λίγο πιο ακριβή. Όπως βλέπουμε όμως, οι τρεις παραπάνω μέθοδοι έχουν μεγαλύτερα σφάλματα από την μέθοδο Runge Kutta 3^{ης} τάξης, που ως μέθοδος ανώτερης τάξης παρέχει καλύτερες προσεγγίσεις της ακριβής λύσεις.

Οι μέθοδοι τέταρτης τάξης βελτιώνουν τις προσεγγίσεις τους και άλλο σε σύγκριση με της μεθόδους τρίτης τάξη και καθώς έχουν τα ίδια περίπου (η μέθοδος Runge Kutta είναι η πιο ακριβής) σφάλματα αποκοπής γι' αυτό και δεν παρουσιάζονται. Στον πίνακα που ακολουθεί παρατηρούμε ότι όσο ανεβαίνει η τάξη της μεθόδου τόσο τα σφάλματα μειώνονται, όμως σε κάθε περίπτωση η μέθοδοι ενός βήματος έχουν μεγαλύτερα σφάλματα από τις πολυβηματικές μεθόδους. Από τις μεθόδους πρόβλεψης- διόρθωσης λαμβάνουμε τις καλύτερες προσεγγίσεις της ακριβούς λύσης.

κομβικά σημεία	**ακριβής λύση**	**Fehlberg5 Error**	**Adams Bashforth4 Error**	**Adams Moulton3 Error**	**Adams4OrderP-C Error**
0	1	0	0	0	0
0,1	1,004837	0,001031	0	0	0
0,2	1,018731	0,001664	0	0	0
0,3	1,040818	0,001962	0	0	0
0,4	1,07032	0,001988	0.000003	0.007415	0,000017
0,5	1,106531	0,001789	0.000004	0.01385	0,0000157
0,6	1,148812	0,00141	0.000006	0.01881	0,0000421
0,7	1,196585	0,000886	0.000008	0.022655	0,000799
0,8	1,249329	0,00025	0.000009	0.025545	0,001281
0,9	1,30657	0,000472	0.00001	0.027625	0,001856
1	1,367879	0,001258	0.000011	0.029029	0,002514

Συμπεράσματα

Οι μέθοδοι της αριθμητικής ανάλυσης που παρουσιάστηκαν έχουν την δυνατότητα να επιλύσουν οποιαδήποτε διαφορική εξίσωση πρώτης τάξης όσο περίπλοκη και εάν είναι. Οι προσεγγίσεις της ακριβής λύσης θα είναι ανάλογα με την μέθοδο που θα ακολουθήσουμε αρκετά ή λιγότερο ακριβείς. Οι μέθοδοι πρώτης και δεύτερης τάξης είναι πολύ απλές στην κατανόηση και στην εφαρμογή αλλά χάνουν στην υπολογιστική ακρίβεια, ενώ το αντίστροφο ισχύει όσο η τάξη και η πολυπλοκότητα των μεθόδων αυξάνεται. Η δυσκολία των αριθμητικών υπολογισμών μπορεί να ξεπεραστεί με την χρήση προγραμμάτων σε κάποια γλώσσα προγραμματισμού ,όπως αυτά που υπάρχουν στο παράρτημα.

Πρόταση για περεταίρω μελέτη

Η μελέτη τα ων διαφορικών εξισώσεων με την χρήση αριθμητικών μεθόδων δεν περιορίζεται σε όσα έχουν μελετηθεί στα πλαίσια αυτής της εργασίας. Ο ενδιαφερόμενος μπορεί με έναυσμα όσα προηγήθηκαν, να μελετήσει διαφορικές εξισώσεις ανώτερης τάξης. Επιπλέον ιδιαίτερο ενδιαφέρον παρουσιάζει η μελέτη συστημάτων διαφορικών εξισώσεων με τις μεθόδους της αριθμητικής ανάλυσης είτε πρόκειται για συστήματα πρώτου βαθμού εξισώσεων είτε μεγαλύτερου. Τα συστήματα αυτά παρουσιάζουν μεγάλο ενδιαφέρον και για την εφαρμογή τους σε συστήματα άλλων επιστημών όπως της φυσικής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] **Richard L.Burden J.Douglas Faires Albert V. Reynolds**

[2] Francis Scheid, 1976,Αριθμητική Ανάλυση, ΕΣΠΙ Αθήνα,422

[3] Νικόλαος Θ. Αποστολάτος,1971,Αριθμητική Ανάλυσις Αθήνα

[4] Μ. Γουσίδου Κουτίτα,1992-1993, Πανεπιστημιακές Παραδόσεις Υπολογιστικών μαθηματικών ΙΙ, Α.Π.Θ. Υπηρεσία Δημοσιευμάτων,σελ.154

[5] Γεώργιος Ακρίβης Βασίλειος Δουγάλης,1998, Εισαγωγή στην Αριθμητική Ανάλυση, Πανεπιστημιακές Εκδόσεις Κρήτης

[6] Μ. Γουσίδου- Κουτίτα,2004, Αριθμητική Ανάλυση, Εκδόσεις Χριστοδουλίδη, σελ157

[7]Τηλέμαχου Ι. Καρβουρίδη,1976,Βασικαί Αρχαί Αριθμητικής Αναλύσεως, Ένωσις Ελλήνων Χημικών,σελ.175

[8] Α. Χατζηδήμου, 1979, Αριθμητική Ανάλυση ΙΙ, Πανεπιστήμιο Ιωαννίνων, σελ.204

[9] Χαραλάμπους Ν. Φραγκάκι, 2008, Μέθοδοι Αριθμητικής Ανάλυσης(Τόμος Ι), Εκδοτικός Οίκος αδελφών Κυριακίδη,σελ.400

[10] Καραμπετάκης Νικόλαος,2002, Εισαγωγή στην Fortran 90/95, Εκδόσεις Ζήτη, σελ381

Χρήσιμες ιστιοσελίδες:

www.ecs.fullerton.edu/~mathews/n2003/rungekutta/RungeKuttaProof.pdf

www.ecs.fullerton.edu

www.astro.auth.gr/~kokkotas/lesson/book_na.pdf

www.edu.physics.uoc.gr/~tety213/notespdf

ΠΑΡΑΡΤΗΜΑ

Στο σημείο αυτό παρουσιάζονται όλοι τα προγράμματα που βασισμένα στους αλγορίθμους που περιέχονται στο κύριο μέρος της εργασίας υπολογίζουν τις προσεγγίσεις όλων των μεθόδων στο γνωστό πρόβλημα αρχικών τιμών. Όλα αυτά τα προγράμματα κατασκευάστηκαν εξ' ολοκλήρου από την συγγραφέα της διπλωματικής. Τα προγράμματα εμφανίζονται χωρίς σχόλια, με αλφαβητική σειρά και ακολουθούνται από το σύνολο των αποτελεσμάτων σε κάθε περίπτωση.

1. **AdamsBashforth4Steps.f90**

```
!
!  FUNCTIONS:
!        AdamsBashforth4Steps      - Entry point of console application.
!
!****************************************************************************
!
!  PROGRAM: AdamsBashforth4Steps
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program AdamsBashforth4Steps
        implicit none
        ! Variables
        real::a,b,y,t,h,w0,w1,w2,w3,t0,t1,t2,t3,w,f
        integer::N,I
        f(t,y)=-y+t+1

      ! Body of AdamsBashforth4Steps
        print*,'Dose ta akra a,b'
        read*,a,b
```

```
print*,'Dose to plithos ton distimaton N'
read*,N
print*,'Dose tin arxiki sinthiki y(0)='
read*,w0
t0=a
        h=(b-a)/N
        t1=a+h; t2=a+2*h; t3=a+3*h
        w1=exp(-t1)+t1 ; w2=exp(-t2)+t2; w3=exp(-t3)+t3
        print*,'t=',t0,'w=',w0,'(initial value)'
        print*,'t=',t1,'w=',w1,'(initial value)'
        print*,'t=',t2,'w=',w2,'(initial value)'
        print*,'t=',t3,'w=',w3,'(initial value)'

         Do I=4,N
         t=t3+h
          w=w3+(h/24)*(55*f(t3,w3)-59*f(t2,w2)+37*f(t1,w1)-9*f(t0,w0))
          print*,'t=',t,'w=',w

                t0=t1
                t1=t2
                t2=t3

                w0=w1
                w1=w2
                w2=w3
                t3=t
                w3=w
        end do
```

```
        end program AdamsBashforth4Steps


! Dose ta akra a,b
!0 1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 w=   1.000000     (initial value)
! t=  0.1000000     w=   1.004837     (initial value)
! t=  0.2000000     w=   1.018731     (initial value)
! t=  0.3000000     w=   1.040818     (initial value)
! t=  0.4000000     w=   1.070323
! t=  0.5000000     w=   1.106535
! t=  0.6000000     w=   1.148818
! t=  0.7000000     w=   1.196593
! t=  0.8000001     w=   1.249338
! t=  0.9000001     w=   1.306580
! t=   1.000000     w=   1.367890
!Press any key to continue
```

2. AdamsFourthOrderPredictorCorrector.f90

```
!
!  FUNCTIONS:
!        AdamsFourthOrderPredictorCorrector        - Entry point of console
application.
!

!*****************************************************************************
!
!  PROGRAM: AdamsFourthOrderPredictorCorrector
!
!  PURPOSE:  Entry point for the console application.
```

```
!
!************************************************************************

        program AdamsFourthOrderPredictorCorrector

        implicit none

        ! Variables
        real::a,b,y,t,h,w0,w1,w2,w3,t0,t1,t2,t3,wp,wc,f
        integer::N,I
        f(t,y)=-y+t+1

        ! Body of AdamsFourthOrderPredictorCorrector
        print*,'Dose ta akra a,b'
        read*,a,b
        print*,'Dose to plithos ton distimaton N'
        read*,N
        print*,'Dose tin arxiki sinthiki y(0)='
        read*,w0
        print*,'Dose tis 3 prwtes times w1,w2,w3'
        read*,w1,w2,w3
                t0=a
                h=(b-a)/N
                t1=a+h; t2=a+2*h; t3=a+3*h
                 Do I=4,N
                 t=t3+h
                  wp=w3+h*(55*f(t3,w3)-59*f(t2,w2)+37*f(t1,w1)-9*f(a,w0))/24
                  wc=w3+h*(9*f(t,wp)+19*f(t3,w3)-5*f(t2,w2)+f(t1,w1))/24
                  print*,'t=',t,'wp=',wp,'wc=',wc

                        t0=t1
                        t1=t2
                        t2=t3
                        w0=w1
                        w1=w2
                        w2=w3
                        t3=t
                        w3=wc
                end do


        end program AdamsFourthOrderPredictorCorrector

!  Dose ta akra a,b
! 0 1
!  Dose to plithos ton distimaton N
! 10
!  Dose tin arxiki sinthiki y(0)=
```

```
! 1
!  Dose tis 3 prwtes times w1,w2,w3
! 1.0048 1.0187 1.0408
! t=  0.4000000     wp=   1.070307     wc=   1.070303
! t=  0.5000000     wp=   1.110269     wc=   1.106374
! t=  0.6000000     wp=   1.156191     wc=   1.148391
! t=  0.7000000     wp=   1.207476     wc=   1.195786
! t=  0.8000001     wp=   1.263635     wc=   1.248048
! t=  0.9000001     wp=   1.324197     wc=   1.304714
! t=   1.000000     wp=   1.388743     wc=   1.365365
! Press any key to continue
```

3. AdamsMoulton3Steps.f90

```
!
!  FUNCTIONS:
!        AdamsMoulton3Steps      - Entry point of console application.
!

!*****************************************************************************
!
!  PROGRAM: AdamsMoulton3Steps
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

        program AdamsMoulton3Steps

        implicit none

        ! Variables
        real::a,b,y,t,h,w0,w1,w2,w3,t0,t1,t2,t3,w,f
        integer::N,I
        f(t,y)=-y+t+1

        ! Body of AdamsMoulton3Steps
        print*,'Dose ta akra a,b'
        read*,a,b
        print*,'Dose to plithos ton distimaton N'
        read*,N
        print*,'Dose tin arxiki sinthiki y(0)='
        read*,w0
        t0=a
                h=(b-a)/N
                t1=a+h; t2=a+2*h; t3=a+3*h
                w1=exp(-t1)+t1 ; w2=exp(-t2)+t2; w3=exp(-t3)+t3
                print*,'t=',t0,'w=',w0,'(initial value)'
```

```
                print*,'t=',t1,'w=',w1,'(initial value)'
                print*,'t=',t2,'w=',w2,'(initial value)'
                print*,'t=',t3,'w=',w3,'(initial value)'

                 Do I=4,N
                 t=t3+h
                  w=w3+(h/24)*(9*f(t3,w3)+19*f(t2,w2)-5*f(t1,w1)+f(t0,w0))
                  print*,'t=',t,'w=',w

                          t0=t1
                          t1=t2
                          t2=t3

                          w0=w1
                          w1=w2
                          w2=w3
                          t3=t
                          w3=w
                end do

        end program AdamsMoulton3Steps


! Dose ta akra a,b
!0 1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 w=   1.000000     (initial value)
! t=  0.1000000     w=   1.004837     (initial value)
! t=  0.2000000     w=   1.018731     (initial value)
! t=  0.3000000     w=   1.040818     (initial value)
! t=  0.4000000     w=   1.062905
! t=  0.5000000     w=   1.092685
! t=  0.6000000     w=   1.130002
! t=  0.7000000     w=   1.173930
! t=  0.8000001     w=   1.223784
! t=  0.9000001     w=   1.278945
! t=   1.000000     w=   1.338850
!Press any key to continue
```

4. EulerMethod.f90

```
!
!  FUNCTIONS:
!        EulerMethod      - Entry point of console application.
!

!****************************************************************************
!
!  PROGRAM: EulerMethod
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program EulerMethod

        implicit none

        ! Variables
REAL::A,B,W,H,T,F,Y
INTEGER::I,N
F(T,Y)=-Y+T+1

        ! Body of EulerMethod

PRINT*,'DOSE TA AKRA A,B'
READ*,A,B
PRINT*,'DOSE TO PLITHOS TON DIASTIMATON N'
READ*,N
PRINT*,'DOSE TIN ARXIKI SINTHIKI y(o)='
READ*,W
        T=A
        H=(B-A)/N
         DO I=0,N

          W=W+H*F(T,W)
          T=A+I*H
    PRINT*,'T=',T,'Y=',W

         END DO

        end program EulerMethod

!  DOSE TA AKRA A,B
! 0 1
!  DOSE TO PLITHOS TON DIASTIMATON N
```

```
! 10
!  DOSE TIN ARXIKI SINTHIKI y(o)=
! 1
!  T=  0.0000000E+00 Y=   1.000000
!  T=  0.1000000     Y=   1.000000
!  T=  0.2000000     Y=   1.010000
!  T=  0.3000000     Y=   1.029000
!  T=  0.4000000     Y=   1.056100
!  T=  0.5000000     Y=   1.090490
!  T=  0.6000000     Y=   1.131441
!  T=  0.7000000     Y=   1.178297
!  T=  0.8000000     Y=   1.230467
!  T=  0.9000000     Y=   1.287421
!  T=   1.000000     Y=   1.348678
! Press any key to continue
```

5. EulerMethodForVoltage.f90

```
!
!  FUNCTIONS:
!        EulerMethodForVoltage      - Entry point of console application.
!

!*****************************************************************************
!
!  PROGRAM: EulerMethodForVoltage
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

        program EulerMethodForVoltage

        implicit none

        ! Variables
REAL::W,H,T,R,L,C,I,E,dE,ddE
INTEGER::J,N
REAL,PARAMETER::PI=3.14
E(T)=(exp(-0.6*PI*T))*sin(2*T-PI)
dE(T)=(exp(-0.6*PI*T))*(-0.6*PI*T*sin(2*T-PI)+2*cos(2*T-PI))
ddE(T)=(0.36*PI**2*T**2-0.6*PI-4)*E(T)-2.4*PI*T*cos(2*T-PI)
I(T)=C*ddE(T)+(1/R)*dE(T)+(1/L)*E(T)

! Body of EulerMethodForVoltage

PRINT*,'Insert the number of repetitions N'
READ*,N
```

```
PRINT*,'Insert the initial condition I(o)='
READ*,W
PRINT*,'Insert the step size h'
READ*,H
PRINT*,'Insert the resistance R,the inductance L and the capacitance C'
READ*,R,L,C

          DO J=1,N
           W=W+H*I(T)
           T=0.1*J
           PRINT*,'T=',T,'I=',W
          END DO
end program EulerMethodForVoltage

! Insert the number of repetitions N
!100
! Insert the initial condition I(o)=
!0
! Insert the step size h
!0.1
! Insert the resistance R,the inductance L and the capacitance C
!1.4,1.7,.3
! T=  0.1000000     I= -0.1426695
! T=  0.2000000     I= -0.2148745
! T=  0.3000000     I= -0.2258282
! T=  0.4000000     I= -0.1890848
! T=  0.5000000     I= -0.1207108
! T=  0.6000000     I= -3.7799977E-02
! T=  0.7000000     I=  4.2719871E-02
! T=  0.8000000     I=  0.1050436
! T=   1.000000     I=  0.1219599
! T=   1.100000     I=  5.6748804E-02
! T=   1.200000     I= -6.5403134E-02
! T=   1.300000     I= -0.2458993
! T=   1.400000     I= -0.4823981
! T=   1.500000     I= -0.7688341
! T=   1.600000     I=  -1.095594
! T=   1.700000     I=  -1.449850
! T=   1.800000     I=  -1.816062
! T=   1.900000     I=  -2.176618
! T=   2.000000     I=  -2.512625
! T=   2.100000     I=  -2.804794
! T=   2.200000     I=  -3.034402
! T=   2.300000     I=  -3.184283
! T=   2.400000     I=  -3.239795
! T=   2.500000     I=  -3.189726
! T=   2.600000     I=  -3.027080
! T=   2.700000     I=  -2.749695
```

```
! T=   2.800000     I=  -2.360671
! T=   2.900000     I=  -1.868554
! T=   3.000000     I=  -1.287270
! T=   3.100000     I= -0.6357987
! T=   3.200000     I=  6.2419586E-02
! T=   3.300000     I=  0.7803143
! T=   3.400000     I=   1.488234
! T=   3.500000     I=   2.155168
! T=   3.600000     I=   2.750095
! T=   3.700000     I=   3.243396
! T=   3.800000     I=   3.608272
! T=   3.900000     I=   3.822094
! T=   4.000000     I=   3.867631
! T=   4.100000     I=   3.734086
! T=   4.200000     I=   3.417882
! T=   4.300000     I=   2.923160
! T=   4.400000     I=   2.261956
! T=   4.500000     I=   1.454025
! T=   4.600000     I=  0.5263156
! T=   4.700000     I= -0.4878924
! T=   4.800000     I=  -1.550168
! T=   4.900000     I=  -2.618381
! T=   5.000000     I=  -3.648375
! T=   5.100000     I=  -4.595789
! T=   5.200000     I=  -5.417958
! T=   5.300000     I=  -6.075809
! T=   5.400000     I=  -6.535674
! T=   5.500000     I=  -6.770925
! T=   5.600000     I=  -6.763367
! T=   5.700000     I=  -6.504317
! T=   5.800000     I=  -5.995306
! T=   5.900000     I=  -5.248372
! T=   6.000000     I=  -4.285908
! T=   6.100000     I=  -3.140061
! T=   6.200000     I=  -1.851703
! T=   6.300000     I= -0.4689902
! T=   6.400000     I=  0.9544289
! T=   6.500000     I=   2.361492
! T=   6.600000     I=   3.694010
! T=   6.700000     I=   4.895075
! T=   6.800000     I=   5.911473
! T=   6.900000     I=   6.696024
! T=   7.000000     I=   7.209725
! T=   7.100000     I=   7.423615
! T=   7.200000     I=   7.320267
! T=   7.300000     I=   6.894833
! T=   7.400000     I=   6.155606
! T=   7.500000     I=   5.124023
```

```
! T=   7.600000     I=   3.834140
! T=   7.700000     I=   2.331547
! T=   7.800000     I=  0.6717921
! T=   7.900000     I=  -1.081665
! T=   8.000000     I=  -2.859874
! T=   8.100000     I=  -4.591104
! T=   8.200000     I=  -6.203735
! T=   8.300000     I=  -7.629221
! T=   8.400001     I=  -8.804990
! T=   8.500000     I=  -9.677164
! T=   8.600000     I=  -10.20300
! T=   8.700000     I=  -10.35288
! T=   8.800000     I=  -10.11189
! T=   8.900001     I=  -9.480708
! T=   9.000000     I=  -8.475985
! T=   9.100000     I=  -7.129992
! T=   9.200000     I=  -5.489653
! T=   9.300000     I=  -3.614942
! T=   9.400001     I=  -1.576705
! T=   9.500000     I=  0.5460033
! T=   9.600000     I=   2.668988
! T=   9.700000     I=   4.706253
! T=   9.800000     I=   6.573481
! T=   9.900001     I=   8.191519
! T=   10.00000     I=   9.489725
!Press any key to continue
```

6. Fehlberg5.f90

```
!
!  FUNCTIONS:
!        Fehlberg5      - Entry point of console application.
!

!****************************************************************************
!
!  PROGRAM: Fehlberg5
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program Fehlberg5

        implicit none

        ! Variables
        REAL::A,B,H,K1,K2,K3,K4,K5,K6,W,Y,F,T
        INTEGER::N,I
```

```
        F(T,Y)=-Y+T+1
        ! Body of Fehlberg5
        PRINT*,'DOSE TA AKRA TOY DIASTIMATOS [XO,XN]'
        READ*,A,B
        PRINT*,'DOSE TO PLITHOS TON DIASTIMATON'
        READ*,N
        PRINT*,'DOSE TIS ARXIKES SYNTHIKES F(0)='
        READ*,W
        H=(B-A)/N
        T=A
         DO I=1,N
                K1=H*F(T,W)
                K2=H*F(T+H/6,W+K1/6)
                K3=H*F(T+(4*H)/6,W+(4*K1)/75+(16/75)*K2)
                K4=H*F(T+(2*H)/3,W+(5*K1)/6-(8*K2)/3+(5*K3)/2)
                K5=H*F(T,W+(361*K1)/320-(18*K2)/5+(407*K3)/128-
(11*K4)/80+(55*K5)/128)
                W=W+(K1+3*K2+3*K3+K4)/8
           T=T+H
          PRINT*,'T=',T,'Y=',W

           END DO
        end program Fehlberg5
 !DOSE TA AKRA TOY DIASTIMATOS [XO,XN]
!0 1
 !DOSE TO PLITHOS TON DIASTIMATON
!10
 !DOSE TIS ARXIKES SYNTHIKES F(0)=
!1
 !T=  0.1000000     Y=   1.003806
 !T=  0.2000000     Y=   1.017067
 !T=  0.3000000     Y=   1.038856
 !T=  0.4000000     Y=   1.068332
 !T=  0.5000000     Y=   1.104742
 !T=  0.6000000     Y=   1.147402
 !T=  0.7000000     Y=   1.195699
 !T=  0.8000001     Y=   1.249079
 !T=  0.9000001     Y=   1.307042
 !T=   1.000000     Y=   1.369137
 !Press any key to continue
```

7. GillMethod.f90

```
!
!  FUNCTIONS:
!        GillMethod      - Entry point of console application.
!
```

```
!*****************************************************************************
!
!  PROGRAM: GillMethod
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

      program GillMethod

      implicit none

      ! Variables
      real::a,b,t,y,w,h,f,k1,k2,k3,k4
      real,parameter::l=1.414213
      integer::I,N
      f(t,y)=-y+t+1

      ! Body of GillMethod

      print*,'Dose ta akra a,b'
      read*,a,b
      print*,'Dose to plithos ton distimaton N'
      read*,N
      print*,'Dose tin arxiki sinthiki y(0)='
      read*,w
            t=a
            h=(b-a)/N
        Do  I=0,N
            k1=h*f(t,w)
            k2=h*f(t+h/2,w+(h*k1)/2)
            k3=h*f(t+h/2,w+(-1/2+l/2)*h*k1+(1-l/2)*h*k2)
            k4=h*f(t+h,w-(l/2)*h*k2+(1+l/2)*h*k3)
           print*,'t=',t,'y=',w
            w=w+h*(k1+(2-l)*k2+(2+l)*k3+k4)/6
            t=t+h
       end do

      end program GillMethod

! Dose ta akra a,b
!0 1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
!t=  0.0000000E+00 y=   1.000000
! t=  0.1000000     y=   1.000498
```

```
! t=  0.2000000     y=   1.001984
! t=  0.3000000     y=   1.004447
! t=  0.4000000     y=   1.007878
! t=  0.5000000     y=   1.012267
! t=  0.6000000     y=   1.017604
! t=  0.7000000     y=   1.023881
! t=  0.8000001     y=   1.031088
! t=  0.9000001     y=   1.039215
! t=   1.000000     y=   1.048254
!Press any key to continue
```

8. HeunMethod.f90

```
!
!  FUNCTIONS:
!          HeunMethod      - Entry point of console application.
!
!*****************************************************************************
!
!  PROGRAM: HeunMethod
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

          program HeunMethod

          implicit none

          ! Variables
          real::a,b,t,y,w,h,f
          integer::I,N
          f(t,y)=-y+t+1

          ! Body of HeunMethod
          print*,'Dose ta akra a,b'
          read*,a,b
          print*,'Dose to plithos ton distimaton N'
          read*,N
          print*,'Dose tin arxiki sinthiki y(0)='
          read*,w
                    t=a
                    h=(b-a)/N
            Do   I=0,N
                      w=w+(h/4)*(f(t,w)+3*f(t+(2/3)*h,w+(2/3)*h*f(t,w)))
                     print*,'t=',t,'y=',w
                      t=t+h
            end do
```

```
        end program HeunMethod

! Dose ta akra a,b
!0 1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 y=   1.000000
! t=  0.1000000     y=   1.010000
! t=  0.2000000     y=   1.029000
! t=  0.3000000     y=   1.056100
! t=  0.4000000     y=   1.090490
! t=  0.5000000     y=   1.131441
! t=  0.6000000     y=   1.178297
! t=  0.7000000     y=   1.230467
! t=  0.8000001     y=   1.287421
! t=  0.9000001     y=   1.348678
! t=  1.000000      y=   1.413811

!Press any key to continue
```

9. IrreversibleChemicalReactionRK4.f90

```
!
!  FUNCTIONS:
!        IrreversibleChemicalReactionRK4            - Entry point of console
application.
!

!*****************************************************************************
!
!  PROGRAM: IrreversibleChemicalReactionRK4
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

        program IrreversibleChemicalReactionRK4

        implicit none

        ! Variables
   real(8)::a,b,t,y,w,h,f,k1,k2,k3,k4,n1,n2,n3,k
        integer::I,N
        f(t,y)=k*((n1-y/2)**2)*((n2-y/2)**2)*((n3-(3*y)/4)**3)

        ! Body of IrreversibleChemicalReactionRK4
```

```
            print*,'Insert the space [a,b]'
        read*,a,b
        print*,'Insert the number of spaces N'
        read*,N
        print*,'Insert the step size h'
        read*,h
        print*,'Insert the velocity constant of the reaction k'
        read*,k
        print*,'Insert the molecules n1,n2,n3'
        read*,n1,n2,n3
        print*,'Insert the initial value y(0)='
        read*,w
            t=a
            h=(b-a)/N
          Do I=1,N+1
             k1=h*f(t,w)
             k2=h*f(t+h/2,w+k1/2)
             k3=h*f(t+h/2,w+k2/2)
             k4=h*f(t+h,w+k3)
            print*,'t=',t,'y=',w
             w=w+(k1+2*k2+2*k3+k4)/6
             t=t+h
           end do
        end program IrreversibleChemicalReactionRK4

!  Insert the space [a,b]
! 0 2
!  Insert the number of spaces N
! 20
!  Insert the step size h
! 0.1
!  Insert the velocity constant of the reaction k
! 0.000000000000622
!  Insert the molecules n1,n2,n3
! 0.001 .001 .0015
!  Insert the initial value y(0)=
! 0
!  t=  0.000000000000000E+000 y=  0.000000000000000E+000
!  t=  0.100000000000000      y=  2.099250000000000E-034
!  t=  0.200000000000000      y=  4.198499999999999E-034
!  t=  0.300000000000000      y=  6.297749999999999E-034
!  t=  0.400000000000000      y=  8.396999999999999E-034
!  t=  0.500000000000000      y=  1.049625000000000E-033
!  t=  0.600000000000000      y=  1.259550000000000E-033
!  t=  0.700000000000000      y=  1.469475000000000E-033
!  t=  0.800000000000000      y=  1.679400000000000E-033
!  t=  0.900000000000000      y=  1.889325000000000E-033
!  t=   1.00000000000000      y=  2.099250000000000E-033
```

```
! t=   1.10000000000000     y=  2.309175000000000E-033
! t=   1.20000000000000     y=  2.519100000000000E-033
! t=   1.30000000000000     y=  2.729025000000000E-033
! t=   1.40000000000000     y=  2.938950000000001E-033
! t=   1.50000000000000     y=  3.148875000000001E-033
! t=   1.60000000000000     y=  3.358800000000001E-033
! t=   1.70000000000000     y=  3.568725000000001E-033
! t=   1.80000000000000     y=  3.778650000000001E-033
! t=   1.90000000000000     y=  3.988575000000001E-033
! t=   2.00000000000000     y=  4.198500000000001E-033
! Press any key to continue
```

10. KuttaMethod.f90

```
!
!  FUNCTIONS:
!        KuttaMethod      - Entry point of console application.
!
!****************************************************************************
!
!  PROGRAM: KuttaMethod
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program KuttaMethod

        implicit none

        ! Variables
        real::a,b,t,y,w,h,f,k1,k2,k3,k4
        integer::I,N
        f(t,y)=-y+t+1

        ! Body of KuttaMethod
        print*,'Dose ta akra a,b'
        read*,a,b
        print*,'Dose to plithos ton distimaton N'
        read*,N
        print*,'Dose tin arxiki sinthiki y(0)='
        read*,w
                t=a
                h=(b-a)/N
           Do I=0,N
                  k1=h*f(t,w)
                  k2=h*f(t+h/3,w+h*k1/3)
                  k3=h*f(t+(2*h)/3,w-(h*k1)/3+h*k2)
```

```
                k4=h*f(t+h,w+h*k1-h*k2+h*k3)
                print*,'t=',t,'y=',w
                w=w+h*(k1+3*k2+3*k3+k4)/8
                t=t+h
            end do
        end program KuttaMethod

! Dose ta akra a,b
!0 1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 y=   1.000000
! t=  0.1000000     y=   1.000498
! t=  0.2000000     y=   1.001987
! t=  0.3000000     y=   1.004455
! t=  0.4000000     y=   1.007894
! t=  0.5000000     y=   1.012294
! t=  0.6000000     y=   1.017645
! t=  0.7000000     y=   1.023938
! t=  0.8000001     y=   1.031163
! t=  0.9000001     y=   1.039312
! t=   1.000000     y=   1.048374
!Press any key to continue
```

11. MidpointMethod.f90

```
!
!  FUNCTIONS:
!        MidpointMethod      - Entry point of console application.
!
!****************************************************************************
!
!  PROGRAM: MidpointMethod
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program MidpointMethod

        implicit none

        ! Variables
        real::a,b,w,h,t,y,k,f
        integer::I,N
        f(t,y)=-y+t+1
```

```
            ! Body of MidpointMethod

            print*,'dose ta akra a,b'
            read*,a,b
            print*,'dose to plithos ton diastimaton N'
            read*,N
            print*,'dose tin arxiki sinthiki y(0)='
            read*,w
              t=a
              h=(b-a)/N
            do I=1,N+1
                 k=w+h*f(t,w)/2
                 print*,'t=',t,'y=',w
                  w=w+h*f(t+h/2,k)
                 t=a+I*h
           end do
         end program MidpointMethod

 ! dose ta akra a,b
 ! dose to plithos ton diastimaton N
 !10
 ! dose tin arxiki sinthiki y(0)=
 !1
 !
 !t=  0.0000000E+00 y=   1.000000
 ! t=  0.1000000     y=   1.005000
 ! t=  0.2000000     y=   1.019025
 ! t=  0.3000000     y=   1.041218
 ! t=  0.4000000     y=   1.070802
 ! t=  0.5000000     y=   1.107076
 ! t=  0.6000000     y=   1.149403
 ! t=  0.7000000     y=   1.197210
 ! t=  0.8000000     y=   1.249975
 ! t=  0.9000000     y=   1.307227
 ! t=   1.000000     y=   1.368541
 !Press any key to continue
```

12. ModifiedEulerMethod.f90

```
!
!  FUNCTIONS:
!         ModifiedEulerMethod      - Entry point of console application.
!

!****************************************************************************
!
!  PROGRAM: ModifiedEulerMethod
```

```
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

      program ModifiedEulerMethod

      implicit none

      ! Variables
      real::a,b,w,h,t,y,k,f
      integer::I,N
      f(t,y)=-y+t+1
      ! Body of ModifiedEulerMethod
      print*,'dose ta akra ton diastimaton a,b'
      read*,a,b
      print*,'dose to plithos ton diastimaton'
      read*,N
      print*,'dose tin  arxiki sinthiki y(0)='
      read*,w
         t=a
         h=(b-a)/N
       Do I=1,N+1
          k=w+h*f(t,w)
          print*,'t=',t,'y=',w
          w=w+(h/2)*(f(t,w)+f(t+h,k))
          t=a+I*h
       end do
      end program ModifiedEulerMethod

! dose ta akra ton diastimaton a,b
!0,1
!dose to plithos ton diastimaton
!10
! dose tin  arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 y=   1.000000
! t=  0.1000000     y=   1.005000
! t=  0.2000000     y=   1.019025
! t=  0.3000000     y=   1.041218
! t=  0.4000000     y=   1.070802
! t=  0.5000000     y=   1.107076
! t=  0.6000000     y=   1.149403
! t=  0.7000000     y=   1.197210
! t=  0.8000000     y=   1.249975
! t=  0.9000000     y=   1.307227
! t=   1.000000     y=   1.368541
```

13. NonconformistsEuler.f90

```
!
!  FUNCTIONS:
!        NonconformistsEuler      - Entry point of console application.
!

!****************************************************************************
!
!  PROGRAM: NonconformistsEuler
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program NonconformistsEuler

        implicit none

        ! Variables
        REAL::B,D,R,P,T,K,L,W,H
        INTEGER::I,N
        P(T)=R*B*(1-T)

        ! Body of NonconformistsEuler
        PRINT*,'Insert the time period [k,l]'
        READ*,K,L
        PRINT*,'INSERT THE STEP SIZE H'
        READ*,H
        PRINT*,'Insert the birth rate B and the death rate D and R'
        READ*,B,D,R
        PRINT*,'Insert the initian value P(0)='
        READ*,W
        T=K; N=(L-K)/H
          DO I=1,N
            W=W+H*P(T)
            T=T+H
            PRINT*,'T=',T,'P=',W
          END DO
        end program NonconformistsEuler



!  Insert the time period [k,l]
! 0,50
!  INSERT THE STEP SIZE H
! 1
!  Insert the birht rate B and the death rate D and R
! .02,.015,.1
```

```
! .01
!  T=   1.000000    P=  1.2000000E-02
!  T=   2.000000    P=  1.2000000E-02
!  T=   3.000000    P=  9.9999998E-03
!  T=   4.000000    P=  5.9999996E-03
!  T=   5.000000    P= -3.6880374E-10
!  T=   7.000000    P= -1.8000001E-02
!  T=   8.000000    P= -3.0000001E-02
!  T=   9.000000    P= -4.4000000E-02
!  T=   10.00000    P= -5.9999999E-02
!  T=   11.00000    P= -7.8000002E-02
!  T=   13.00000    P= -0.1200000
!  T=   14.00000    P= -0.1440000
!  T=   15.00000    P= -0.1700000
!  T=   16.00000    P= -0.1980000
!  T=   17.00000    P= -0.2280000
!  T=   18.00000    P= -0.2600000
!  T=   20.00000    P= -0.3300000
!  T=   21.00000    P= -0.3680000
!  T=   22.00000    P= -0.4080000
!  T=   23.00000    P= -0.4500000
!  T=   24.00000    P= -0.4940000
!  T=   25.00000    P= -0.5400000
!  T=   26.00000    P= -0.5879999
!  T=   27.00000    P= -0.6380000
!  T=   28.00000    P= -0.6899999
!  T=   29.00000    P= -0.7440000
!  T=   30.00000    P= -0.8000000
!  T=   31.00000    P= -0.8580000
!  T=   32.00000    P= -0.9180000
!  T=   33.00000    P= -0.9800000
!  T=   34.00000    P=  -1.044000
!  T=   35.00000    P=  -1.110000
!  T=   36.00000    P=  -1.178000
!  T=   37.00000    P=  -1.248000
!  T=   38.00000    P=  -1.320000
!  T=   39.00000    P=  -1.394000
!  T=   40.00000    P=  -1.470000
!  T=   41.00000    P=  -1.548000
!  T=   42.00000    P=  -1.628000
!  T=   43.00000    P=  -1.710000
!  T=   44.00000    P=  -1.794000
!  T=   45.00000    P=  -1.880000
!  T=   46.00000    P=  -1.968000
!  T=   47.00000    P=  -2.058000
!  T=   48.00000    P=  -2.150000
!  T=   49.00000    P=  -2.244000
!  T=   50.00000    P=  -2.340000
```

14. RungeKuttaFehlberg.f90

```
!
!  FUNCTIONS:
!        RungeKuttaFehlberg      - Entry point of console application.
!

!*****************************************************************************
!
!  PROGRAM: RungeKuttaFehlberg
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

        program RungeKuttaFehlberg

        implicit none

        ! Variables

        real::a,b,w,t,y,TOL,hmax,hmin,h,k1,k2,k3,k4,k5,k6,R,d,f
        integer::N
        f(t,y)=-y+t+1
        ! Body of RungeKuttaFehlberg
        print*,'Dose ta akra a,b'
        read*,a,b
        print*,'Dose to plithos N,ton diastimaton'
        read*,N
        print*,'dose tolerance,maximun step size and minimum step size'
        read*,TOL,hmax,hmin
        print*,'Dose tin arxiki sinthiki'
        read*,w
        t=a
        h=hmax
   do while (t.LE.b)
                k1=h*f(t,w)
                k2=h*f(t+h/4.,w+k1/4.)
                k3=h*f(t+(3.*h)/8.,w+(3.*k1)/32.+(9.*k2)/32.)
                k4=h*f(t+(12.*h)/13.,w+(1932.*k1)/2197.-
(7200.*k2)/2197.+(7296.*k3)/2197.)
                k5=h*f(t+h,w+(439.*k1)/216.-8.*k2+(3680.*k3)/513.-
(845.*k4)/4104.)
                k6=h*f(t+h/2.,w-(8.*k1)/27.+2.*k2-
(3544.*k3)/2565.+(1859.*k4)/4104.-(11.*k5)/40.)
                R=abs(k1/360.-(128.*k3)/4275.-
(2197.*k4)/75240.+k5/50.+(2.*k6)/55.)/h
                d=(.84)*(TOL/R)**(0.25)
```

```
                    If(R.LE.TOL) then
                     t=t+h
                     w=w+(25.*k1)/216.+(1408.*k3)/2565.+(2197.*k4)/4104.-(k5/5.)
                     print*,'t=',t,'y=',w,'h=',h,'R=',R,'d=',d

                    end if
                    If(d.LE.0.1) then
                     h=0.1*h
                    else if(d.GE.4) then
                     h=4*h
                    else
                     h=d*h
                    end if
               If(h.GT.hmax) then
                     h=hmax
                    end if
                    If(h.LT.hmin) then
                     print*,'minimun h exceeded-procedure completed
unsuccessfully'
                    end if
   end do
         end program RungeKuttaFehlberg

! Dose ta akra a,b
!0 1
 !Dose to plithos N,ton diastimaton
!10
 !dose tolerance,maximun step size and minimum step size
!0.00005 0.1 0.02
 !Dose tin arxiki sinthiki
!1
 !t=  0.1000000    y=   1.004837    h=  0.1000000    R=  1.3087417E-07 d=
 !  3.713712
 !t=  0.2000000    y=   1.018731    h=  0.1000000    R=  1.2301388E-07 d=
 !  3.771666
 !t=  0.3000000    y=   1.040818    h=  0.1000000    R=  1.0627485E-07 d=
 !  3.912138
 !t=  0.4000000    y=   1.070320    h=  0.1000000    R=  1.0275755E-07 d=
 !  3.945194
 !t=  0.5000000    y=   1.106531    h=  0.1000000    R=  8.8467623E-08 d=
 !  4.095677
 !t=  0.6000000    y=   1.148812    h=  0.1000000    R=  8.1265398E-08 d=
 !  4.183554
 !t=  0.7000000    y=   1.196585    h=  0.1000000    R=  7.4158727E-08 d=
 !  4.280369
 !t=  0.8000001    y=   1.249329    h=  0.1000000    R=  6.5205256E-08 d=
 !  4.467321
 !t=  0.9000001    y=   1.306570    h=  0.1000000    R=  5.9072487E-08 d=
```

```
 !  4.530806
 !t=   1.000000     y=   1.367880     h=  0.1000000     R=  5.2529959E-08 d=
 ! 4.665734
```

15. RungeKuttaOrder3.f90

```
!
!  FUNCTIONS:
!        RungeKuttaOrder3      - Entry point of console application.
!

!****************************************************************************
!
!  PROGRAM: RungeKuttaOrder3
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program RungeKuttaOrder3

        implicit none

        ! Variables
        real::a,b,t,y,w,h,f,k1,k2,k3
        integer::I,N
        f(t,y)=-y+t+1


        ! Body of RungeKuttaOrder3

        print*,'Dose ta akra a,b'
        read*,a,b
        print*,'Dose to plithos ton distimaton N'
        read*,N
        print*,'Dose tin arxiki sinthiki y(0)='
        read*,w
                t=a
                h=(b-a)/N
           Do I=0,N
                k1=h*f(t,w)
                k2=h*f(t+h/2,w+k1/2)
                k3=h*f(t+h,w-k1+2*k2)
               print*,'t=',t,'y=',w
                w=w+(k1+4*k2+k3)/6
                t=t+h
           end do
```

```
        end program RungeKuttaOrder3

! Dose ta akra a,b
!0,1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 y=   1.000000
! t=  0.1000000     y=   1.004833
! t=  0.2000000     y=   1.187233
! t=  0.3000000     y=   1.040808
! t=  0.4000000     y=   1.070308
! t=  0.5000000     y=   1.106517
! t=  0.6000000     y=   1.148797
! t=  0.7000000     y=   1.196570
! t=  0.8000001     y=   1.249313
! t=  0.9000001     y=   1.306553
! t=   1.000000     y=   1.367863
```

16. RungeKuttaOrder4.f90

```
!
!  FUNCTIONS:
!        RungeKuttaOrder4      - Entry point of console application.
!

!****************************************************************************
!
!  PROGRAM: RungeKuttaOrder4
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************

        program RungeKuttaOrder4

        implicit none

        ! Variables

        real::a,b,t,y,w,h,f,k1,k2,k3,k4
        integer::I,N
        f(t,y)=-y+t+1

        ! Body of RungeKuttaOrder4
                print*,'Dose ta akra a,b'
                read*,a,b
```

```
            print*,'Dose to plithos ton distimaton N'
            read*,N
            print*,'Dose tin arxiki sinthiki y(0)='
            read*,w
                    t=a
                    h=(b-a)/N
                Do I=0,N
                    k1=h*f(t,w)
                    k2=h*f(t+h/2,w+k1/2)
                    k3=h*f(t+h/2,w+k2/2)
                    k4=h*f(t+h,w+k3)
                    print*,'t=',t,'y=',w
                    w=w+(k1+2*k2+2*k3+k4)/6
                    t=t+h
                end do
        end program RungeKuttaOrder4

! Dose ta akra a,b
!0,1
! Dose to plithos ton distimaton N
!10
! Dose tin arxiki sinthiki y(0)=
!1
! t=  0.0000000E+00 y=   1.000000
! t=  0.1000000     y=   1.004838
! t=  0.2000000     y=   1.018731
! t=  0.3000000     y=   1.040818
! t=  0.4000000     y=   1.070320
! t=  0.5000000     y=   1.106531
! t=  0.6000000     y=   1.148812
! t=  0.7000000     y=   1.196586
! t=  0.8000001     y=   1.249329
! t=  0.9000001     y=   1.306570
! t=   1.000000     y=   1.367880
!Press any key to continue
```

17. SpruceBudworm.f90

```
!
!  FUNCTIONS:
!          SpruceBudworm      - Entry point of console application.
!
!****************************************************************************
!
!  PROGRAM: SpruceBudworm
!
!  PURPOSE:  Entry point for the console application.
!
!****************************************************************************
```

```
        program SpruceBudworm

        implicit none

        ! Variables

        real::l,m,t,x,w,h,f,k1,k2,k3,k4,r,K,b,p,a
        integer::l,N
        f(x)=r*x*(1-(x/K))- (b*p*(x**2))/(x**2+a**2)

    ! Body of SpruceBudworm
        print*,'Dose ta akra l,m tou [l,m]'
        read*,l,m
        print*,'Dose to plithos ton distimaton N'
        read*,N
        print*,'dose to bhma h'
        read*,h
        print*,'Dose tin arxiki sinthiki x(0)='
        read*,w
        print*,'Dose tiw times tvn parametrwn r,K,b,p,a'
        read*,r,K,b,p,a
                t=l
          Do l=0,N+1
                k1=h*f(w)
                k2=h*f(w+k1/2)
                k3=h*f(w+k2/2)
                k4=h*f(w+k3)
                 print*,'t=',t,'x=',w
                w=w+(k1+2*k2+2*k3+k4)/6
                t=t+h
          end do
        end program SpruceBudworm

! Dose ta akra l,m tou [l,m]
0 20
 Dose to plithos ton distimaton N
100
 dose to bhma h
0.2
 Dose tin arxiki sinthiki x(0)=
5
 Dose tiw times tvn parametrwn r,K,b,p,a
0.8 43 1.4 1.6 1.2
 t=  0.0000000E+00 x=   5.000000
 t=  0.2000000     x=   5.300055
 t=  0.4000000     x=   5.635895
 t=  0.6000000     x=   6.011316
```

t= 0.8000000 x= 6.430271
t= 1.000000 x= 6.896798
t= 1.200000 x= 7.414923
t= 1.400000 x= 7.988538
t= 1.600000 x= 8.621247
t= 1.800000 x= 9.316194
t= 2.000000 x= 10.07586
t= 2.200000 x= 10.90185
t= 2.400000 x= 11.79467
t= 2.600000 x= 12.75355
t= 2.800000 x= 13.77621
t= 3.000000 x= 14.85877
t= 3.200001 x= 15.99565
t= 3.400001 x= 17.17963
t= 3.600001 x= 18.40193
t= 3.800001 x= 19.65243
t= 4.000000 x= 20.92000
t= 4.200000 x= 22.19287
t= 4.400000 x= 23.45906
t= 4.600000 x= 24.70683
t= 4.800000 x= 25.92510
t= 5.000000 x= 27.10389
t= 5.199999 x= 28.23452
t= 5.399999 x= 29.30989
t= 5.599999 x= 30.32457
t= 5.799999 x= 31.27479
t= 5.999999 x= 32.15837
t= 6.199998 x= 32.97462
t= 6.399998 x= 33.72409
t= 6.599998 x= 34.40843
t= 6.799998 x= 35.03011
t= 6.999998 x= 35.59227
t= 7.199997 x= 36.09848
t= 7.399997 x= 36.55259
t= 7.599997 x= 36.95860
t= 7.799997 x= 37.32050
t= 7.999997 x= 37.64220
t= 8.199997 x= 37.92750
t= 8.399997 x= 38.17997
t= 8.599997 x= 38.40298
t= 8.799996 x= 38.59962
t= 8.999996 x= 38.77277
t= 9.199996 x= 38.92504
t= 9.399996 x= 39.05878
t= 9.599996 x= 39.17615
t= 9.799995 x= 39.27905
t= 9.999995 x= 39.36919
t= 10.20000 x= 39.44811

t= 10.39999 x= 39.51716
t= 10.59999 x= 39.57755
t= 10.79999 x= 39.63033
t= 10.99999 x= 39.67646
t= 11.19999 x= 39.71674
t= 11.39999 x= 39.75192
t= 11.59999 x= 39.78263
t= 11.79999 x= 39.80943
t= 11.99999 x= 39.83282
t= 12.19999 x= 39.85323
t= 12.39999 x= 39.87103
t= 12.59999 x= 39.88655
t= 12.79999 x= 39.90009
t= 12.99999 x= 39.91190
t= 13.19999 x= 39.92219
t= 13.39999 x= 39.93117
t= 13.59999 x= 39.93899
t= 13.79999 x= 39.94581
t= 13.99999 x= 39.95176
t= 14.19999 x= 39.95694
t= 14.39999 x= 39.96146
t= 14.59999 x= 39.96540
t= 14.79999 x= 39.96883
t= 14.99999 x= 39.97182
t= 15.19999 x= 39.97443
t= 15.39999 x= 39.97671
t= 15.59999 x= 39.97869
t= 15.79999 x= 39.98041
t= 15.99999 x= 39.98191
t= 16.19999 x= 39.98323
t= 16.39999 x= 39.98437
t= 16.59999 x= 39.98537
t= 16.79999 x= 39.98623
t= 16.99999 x= 39.98699
t= 17.19999 x= 39.98765
t= 17.39999 x= 39.98822
t= 17.59999 x= 39.98872
t= 17.80000 x= 39.98916
t= 18.00000 x= 39.98954
t= 18.20000 x= 39.98987
t= 18.40000 x= 39.99016
t= 18.60000 x= 39.99041
t= 18.80000 x= 39.99063
t= 19.00000 x= 39.99082
t= 19.20000 x= 39.99099
t= 19.40000 x= 39.99113
t= 19.60000 x= 39.99126
t= 19.80000 x= 39.99137

```
 t=  20.00000     x=  39.99147
 t=  20.20000     x=  39.99155
!Press any key to continue
```

18. TaylorOrder2.f90

```
!
!  FUNCTIONS:
!        TaylorOrder2      - Entry point of console application.
!
!*****************************************************************************
!
!  PROGRAM: TaylorOrder2
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

        program TaylorOrder2

        implicit none

        ! Variables
        REAL::A,B,W,H,T,Y,F,DF,K
        INTEGER::I,N
         F(T,Y)=-Y+T+1
         DF(T,Y)=Y-T
         K=F(T,Y)+(H/2)*DF(T,Y)

        ! Body of TaylorOrder2

        PRINT*,'DOSE TA AKRA A,B'
        READ*,A,B
        PRINT*,'DOSE TO PLITHOS TON DIASTIMATON N'
        READ*,N
        PRINT*,'DOSE TIN ARXIKI SINTHIKI Y(A)='
        READ*,W
          H=(B-A)/N
          T=A
         DO I=1,N+1
              K=F(T,W)+(H/2)*DF(T,W)
              PRINT*,'T=',T,'W=',W
              W=W+H*K
             T=A+I*H
         END DO
        end program TaylorOrder2

!  DOSE TA AKRA A,B
! 0 1
```

```
!  DOSE TO PLITHOS TON DIASTIMATON N
! 10
!  DOSE TIN ARXIKI SINTHIKI Y(A)=
! 1
!  T=  0.0000000E+00 W=   1.000000
!  T=  0.1000000     W=   1.005000
!  T=  0.2000000     W=   1.019025
!  T=  0.3000000     W=   1.041218
!  T=  0.4000000     W=   1.070802
!  T=  0.5000000     W=   1.107076
!  T=  0.6000000     W=   1.149403
!  T=  0.7000000     W=   1.197210
!  T=  0.8000000     W=   1.249975
!  T=  0.9000000     W=   1.307227
!  T=   1.000000     W=   1.368541
! Press any key to continue
```

19. TaylorOrder4.f90

```
!
!  FUNCTIONS:
!          TaylorOrder4      - Entry point of console application.
!
!*****************************************************************************
!
!  PROGRAM: TaylorOrder4
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

          program TaylorOrder4

          implicit none

          ! Variables
          real::a,b,w,h,t,y,f,df,ddf,dddf,K
           integer::I,N
           f(t,y)=-y+t+1
           df(t,y)=y-t
           ddf(t,y)=t-y
           dddf(t,y)=y-t
            K=f(t,y)+h/2
          ! Body of TaylorOrder4
            print*,'dose ta akra a,b'
            read*,a,b
            print*,'dose to plithos ton diastimaton N'
            read*,N
            print*,'dose tin arxiki sinthiki y(a)='
```

```
      read*,w
       h=(b-a)/n
       t=a
      do I=1,N+1
       K=f(t,w)+(h/2)*df(t,w)+(h**2)/6*ddf(t,w)+(h**3)/2*dddf(t,w)
       print*,'t=',t,'y=',w
            w=w +h*K
       t=a+I*h
      end do
     end program TaylorOrder4

! dose ta akra a,b
! 0,1
! dose to plithos ton diastimaton N
! 10
! dose tin arxiki sinthiki y(a)=
! 1
!  t=  0.0000000E+00 y=   1.000000
!  t=  0.1000000     y=   1.004883
!  t=  0.2000000     y=   1.018814
!  t=  0.3000000     y=   1.040931
!  t=  0.4000000     y=   1.070456
!  t=  0.5000000     y=   1.106685
!  t=  0.6000000     y=   1.148979
!  t=  0.7000000     y=   1.196762
!  t=  0.8000000     y=   1.249511
!  t=  0.9000000     y=   1.306755
!  t=   1.000000     y=   1.368066
! Press any key to continue
```

20. VerticalShotTaylor2.f90

```
!
!  FUNCTIONS:
!        VerticalShotTaylor2      - Entry point of console application.
!
!*****************************************************************************
!
!  PROGRAM: VerticalShotTaylor2
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

      program VerticalShotTaylor2

      implicit none
```

```
        ! Variables

        REAL::a,b,w,h,t,u,du,ddu,k,m,Fb,Fr
        INTEGER::I,N
        REAL,PARAMETER::g=9.8
        du(t)=-g-(k*t**2)/m
        ddu(t)=-1-(2*k*u*du(t))/m
        ! Body of VerticalShotTaylor2
        PRINT*,'Insert the time period [a,b]'
        READ*,a,b
        PRINT*,'Insert the step size h'
        READ*,h
        PRINT*,'Insert the number of repetitions'
        READ*,N
        PRINT*,'Insert the projectile mass m'
        READ*,m
        PRINT*,'Insert the parameter k of air resistance'
        READ*,k
        PRINT*,'Insert the initial value u(0)='
        READ*,w
        Fb=-m*g
        t=a
          Do I=1,N
            w=w+h*(du(t)+(h/2)*ddu(t))
            u=w
             Fr=-k*u**2
             t=t+h
             PRINT*,'t=',t,'u=',u,'Fb=',Fb,'Fr=',Fr
          END DO
      end program VerticalShotTaylor2

! Insert the time period [a,b]
!         .1,1
! Insert the step size h
!         .1
! Insert the number of repetitions
!         10
! Insert the projectile mass m
!   .11
! Insert the parameter k of air resistance
!   .002
! Insert the initial value u(0)=
!   8
! t=  0.1000000     u=   7.015000     Fb=  -1.078000     Fr= -9.8420449E-02
! t=  0.2000000     u=   6.042481     Fb=  -1.078000     Fr= -7.3023170E-02
! t=  0.3000000     u=   5.068176     Fb=  -1.078000     Fr= -5.1372822E-02
! t=  0.4000000     u=   4.092045     Fb=  -1.078000     Fr= -3.3489663E-02
! t=  0.5000000     u=   3.114047     Fb=  -1.078000     Fr= -1.9394582E-02
```

```
! t=  0.6000000     u=   2.134144     Fb=  -1.078000     Fr= -9.1091422E-03
! t=  0.7000000     u=   1.152295     Fb=  -1.078000     Fr= -2.6555660E-03
! t=  0.8000001     u=  0.1684587     Fb=  -1.078000     Fr= -5.6756700E-05
! t=  0.9000001     u= -0.8174044     Fb=  -1.078000     Fr= -1.3362999E-03
! t=   1.000000     u=  -1.805336     Fb=  -1.078000     Fr= -6.5184748E-03
!Press any key to continue
```

21. multiprogram.f90

(το πρόγραμμα που ακολουθεί είναι ένας συνδθασμός του προηγούμενου προγράμματος με τα προγράμματα Runge Kutta Order3 & Runge Kutta Order 4. Ο χρήστης μπορεί να υπολογίσει ένα από τα εξής:

- **τις 3 πρώτες τιμές από Runge Kutta3**
- **τις 3 πρώτες τιμές από Runge Kutta4**
- **προσεγγιστικές λύσεις από το πρόγραμμα πρόβλεψης – διόρθωσης με αρχικές τιμές υπολογισμένες από Runge Kutta 3**
- **προσεγγιστικές λύσεις από το πρόγραμμα πρόβλεψης – διόρθωσης με αρχικές τιμές υπολογισμένες από Runge Kutta 4)**

```
!
!  FUNCTIONS:
!        RungeKuttaOrder4      - Entry point of console application.
!

!*****************************************************************************
!
!  PROGRAM: Diplomatiki
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************

      program multiprogram

            implicit none

            ! Variables
            real::a,b,w,results(3)
            integer::I,N,epilogi,AdamsN

            ! Body of multiprogram
            print*,'Dose ta akra a,b'
            read*,a,b
            N=3
            print*,'Dose tin arxiki sinthiki y(0)='
```

```
                read*,w
                print*, ' '
                print*, '/******-Menu-******/'
                print*, '1. RungeKuttaOrder3'
                print*, '2. RungeKuttaOrder4'
                print*, '3. AdamsFourthOrderPredictorCorrector (by
RungeKuttaOrder3)'
                print*, '4. AdamsFourthOrderPredictorCorrector (by
RungeKuttaOrder4)'
                print*, '0. Exit'
                print*, '/******************/'
                read*,epilogi
                do while (epilogi.NE.0)

                        if(epilogi==1) then
                                call RungeKuttaOrder3(a,b,N,w,results)
                        elseif(epilogi==2) then
                                call RungeKuttaOrder4(a,b,N,w,results)
                        elseif(epilogi==3) then
                                call RungeKuttaOrder3(a,b,N,w,results)
                                print*,' '
                                print*,'Dose to plithos ton diastimaton N
(>4) gia tin AdamsFourthOrderPredictorCorrector'
                                read*,AdamsN
                                call
AdamsFourthOrderPredictorCorrector(a,b,AdamsN,w,results(1),result
s(2),results(3))
                        elseif(epilogi==4) then
                                call RungeKuttaOrder4(a,b,N,w,results)
                                print*,' '
                                print*,'Dose to plithos ton diastimaton N
(>4) gia tin AdamsFourthOrderPredictorCorrector'
                                read*,AdamsN
                                call
AdamsFourthOrderPredictorCorrector(a,b,AdamsN,w,results(1),result
s(2),results(3))
                        end if
                        print*, ' '
                        print*, '/******-Rsults-******/'
                        DO I=1,N
                                print*,results(I)
                        end do
                        print*, '/******************/'
                        print*, ' '
                        print*, '/******-Menu-******/'
                        print*, '1. RungeKuttaOrder3'
                        print*, '2. RungeKuttaOrder4'
```

```
                print*,  '3.  AdamsFourthOrderPredictorCorrector
(by RungeKuttaOrder3)'
                print*,  '4.  AdamsFourthOrderPredictorCorrector
(by RungeKuttaOrder4)'
                print*, '0. Exit'
                print*, '/******************/'
                read*,epilogi
        enddo

    end program multiprogram

    subroutine RungeKuttaOrder3(a,b,N,w,results)

        implicit none

        ! Variables
        real::a,b,t,y,w,h,f,k1,k2,k3
        real,INTENT(OUT) :: results(3)
        integer::I,N
        f(t,y)=-y+t+1

        t=a
        h=(b-a)/N
        Do I=0,N-1
            k1=h*f(t,w)
            k2=h*f(t+h/2,w+k1/2)
            k3=h*f(t+h,w-k1+2*k2)
            print*,'t=',t,'y=',w
            results(I+1)=w
            w=w+(k1+4*k2+k3)/6
            t=t+h
        end do

    end subroutine RungeKuttaOrder3

    subroutine RungeKuttaOrder4(a,b,N,w,results)

        implicit none

        ! Variables
        real::a,b,t,y,w,h,f,k1,k2,k3,k4
        real,INTENT(OUT) :: results(3)
        integer::I,N
        f(t,y)=-y+t+1

        t=a
        h=(b-a)/N
        Do I=0,N-1
```

```
                k1=h*f(t,w)
                k2=h*f(t+h/2,w+k1/2)
                k3=h*f(t+h/2,w+k2/2)
                k4=h*f(t+h,w+k3)
                print*,'t=',t,'y=',w
                results(I+1)=w
                w=w+(k1+2*k2+2*k3+k4)/6
                t=t+h
        end do

    end subroutine RungeKuttaOrder4

    subroutine
AdamsFourthOrderPredictorCorrector(a,b,N,w0,w1,w2,w3)

        implicit none

        ! Variables
        real::a,b,y,t,h,w0,w1,w2,w3,t0,t1,t2,t3,wp,wc,f
        integer::N,I
        f(t,y)=-y+t+1

        t0=a
        h=(b-a)/N
        t1=a+h; t2=a+2*h; t3=a+3*h
        Do I=4,N
            t=t3+h
            wp=w3+h*(55*f(t3,w3)-59*f(t2,w2)+37*f(t1,w1)-
9*f(a,w0))/24
            wc=w3+h*(9*f(t,wp)+19*f(t3,w3)-
5*f(t2,w2)+f(t1,w1))/24
            print*,'t=',t,'wp=',wp,'wc=',wc
            t0=t1
            t1=t2
            t2=t3

            w0=w1
            w1=w2
            w2=w3
            t3=t
            w3=wc
        end do

    end subroutine AdamsFourthOrderPredictorCorrector



 Dose ta akra a,b
```

```
0 1
 Dose tin arxiki sinthiki y(0)=
1

 /******-Menu-******/
 1. RungeKuttaOrder3
 2. RungeKuttaOrder4
 3. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder3)
 4. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder4)
 0. Exit
 /******************/
1
 t=  0.0000000E+00 y=   1.000000
 t=  0.3333333     y=   1.049383
 t=  0.6666667     y=   1.179393

 /******-Rsults-******/
   1.000000
   1.049383
   1.179393
 /******************/

 /******-Menu-******/
 1. RungeKuttaOrder3
 2. RungeKuttaOrder4
 3. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder3)
 4. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder4)
 0. Exit
 /******************/
2
 t=  0.0000000E+00 y=   1.367138
 t=  0.3333333     y=   1.312975
 t=  0.6666667     y=   1.368642

 /******-Rsults-******/
   1.367138
   1.312975
   1.368642
 /******************/

 /******-Menu-******/
 1. RungeKuttaOrder3
 2. RungeKuttaOrder4
 3. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder3)
 4. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder4)
 0. Exit
 /******************/
3
```

```
 t=  0.0000000E+00 y=   1.503010
 t=  0.3333333     y=   1.409563
 t=  0.6666667     y=   1.437300

 Dose to plithos ton diastimaton N (>4) gia tin
AdamsFourthOrderPredictorCorrect

 or
10
 t=  0.4000000     wp=   1.415915     wc=   1.428520
 t=  0.5000000     wp=   1.442293     wc=   1.430414
 t=  0.6000000     wp=   1.447563     wc=   1.441661
 t=  0.7000000     wp=   1.472842     wc=   1.461146
 t=  0.8000001     wp=   1.503756     wc=   1.488156
 t=  0.9000001     wp=   1.541453     wc=   1.521973
 t=   1.000000     wp=   1.585328     wc=   1.561949

 /******-Rsults-******/
   1.488156
   1.521973
   1.561949
 /******************/

 /******-Menu-******/
 1. RungeKuttaOrder3
 2. RungeKuttaOrder4
 3. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder3)
 4. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder4)
 0. Exit
 /******************/
4
 t=  0.0000000E+00 y=   1.461146
 t=  0.3333333     y=   1.380338
 t=  0.6666667     y=   1.416912

 Dose to plithos ton diastimaton N (>4) gia tin
AdamsFourthOrderPredictorCorrect

 or
10
 t=  0.4000000     wp=   1.398936     wc=   1.409949
 t=  0.5000000     wp=   1.425901     wc=   1.413624
 t=  0.6000000     wp=   1.432103     wc=   1.426479
 t=  0.7000000     wp=   1.459110     wc=   1.447408
 t=  0.8000001     wp=   1.491326     wc=   1.475725
 t=  0.9000001     wp=   1.530205     wc=   1.510725
 t=   1.000000     wp=   1.575150     wc=   1.551771
```

```
/******-Rsults-******/
  1.475725
  1.510725
  1.551771
/******************/

/******-Menu-******/
1. RungeKuttaOrder3
2. RungeKuttaOrder4
3. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder3)
4. AdamsFourthOrderPredictorCorrector (by RungeKuttaOrder4)
0. Exit
/******************/
0
Press any key to continue
```