ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

"ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΘΕΩΡΙΑ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΕΛΕΓΧΟΥ"

ΑΡΙΘΜΗΤΙΚΗ ΔΙΕΡΕΥΝΗΣΗ ΜΕΘΟΔΩΝ ΕΠΙΛΥΣΗΣ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΜΕ ΕΜΦΑΣΗ ΣΤΙΣ ΜΕΘΟΔΟΥΣ FEHLBERG- ΕΦΑΡΜΟΓΕΣ ΜΕ ΠΡΟΓΡΑΜΜΑΤΑ ΣΕ MATLAB

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΑΔΗΣ Α. ΝΙΚΟΛΑΟΣ

Επιβλέπων: Γουσίδου-Κουτίτα Μαρία
Αναπληρώτρια Καθηγήτρια Α.Π.Θ.

Θεσσαλονίκη, Φεβρουάριος 2011

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

"ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΘΕΩΡΙΑ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΕΛΕΓΧΟΥ"

ΑΡΙΘΜΗΤΙΚΗ ΔΙΕΡΕΥΝΗΣΗ ΜΕΘΟΔΩΝ ΕΠΙΛΥΣΗΣ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΜΕ ΕΜΦΑΣΗ ΣΤΙΣ ΜΕΘΟΔΟΥΣ FEHLBERG- ΕΦΑΡΜΟΓΕΣ ΜΕ ΠΡΟΓΡΑΜΜΑΤΑ ΣΕ MATLAB

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΑΔΗΣ Α. ΝΙΚΟΛΑΟΣ

Επιβλέπων: Γουσίδου-Κουτίτα Μαρία
Αναπληρώτρια Καθηγήτρια Α.Π.Θ.

Εγκρίθηκε από την τριμελή επιτροπή την.....

.....	.....	.....
Μ. Γουσίδου-Κουτίτα	Ν. Καραμπετάκης	Γ. Ραχώνης
Αν. Καθηγήτρια Α.Π.Θ.	Επ.Καθηγητής Α.Π.Θ.	Επ.Καθηγητής Α.Π.Θ.

Θεσσαλονίκη, Φεβρουάριος 2011

.....

Γεωργιάδης Α. Νικόλαος

Πτυχιούχος Μαθηματικός Α.Π.Θ.

ΠΕΡΙΛΗΨΗ

Πολλά φυσικά φαινόμενα περιγράφονται μαθηματικά από διαφορικές εξισώσεις είτε είναι συνήθεις διαφορικές εξισώσεις είτε διαφορικές εξισώσεις με μερικές παραγώγους. Δυστυχώς λίγες διαφορικές εξισώσεις λύνονται αναλυτικά γι αυτό ήδη από τις αρχές του αιώνα έγιναν προσπάθειες για να βρεθούν αριθμητικές μέθοδοι επίλυσης των διαφορικών εξισώσεων. Το κύριο μέρος της έρευνας στην αριθμητική επίλυση διαφορικών εξισώσεων ξεκίνησε από την δημιουργία των πρώτων ηλεκτρονικών υπολογιστών.

Η αριθμητική επίλυση των διαφορικών εξισώσεων συνίσταται στην κατασκευή αλγορίθμων και στην συνέχεια η υλοποίηση τους με λογισμικό Η/Υ. Μεγάλο μέρος της έρευνας στα εφαρμοσμένα μαθηματικά αποτελεί η κατασκευή τέτοιων αλγορίθμων.

Οι αριθμητικές μέθοδοι επίλυσης συνήθων διαφορικών εξισώσεων επιλύουν προβλήματα αρχικών τιμών πρώτης και δεύτερης τάξης (όλα τα άλλα προβλήματα ανάγονται σε αυτά).

Δύο μεγάλες ομάδες μεθόδων επίλυσης διαφορικών εξισώσεων είναι οι μέθοδοι απλού βήματος και οι πολυβηματικές μέθοδοι (Multistep methods). Υπάρχουν αρκετές μέθοδοι απλού βήματος. Μεταξύ αυτών είναι και οι μέθοδοι Runge Kutta και Runge Kutta Nystrom. Η μεγάλη ακρίβεια και η ευκολία εφαρμογής των αλγορίθμων Runge-Kutta και Runge Kutta Nystrom τις κατατάσσουν στις πιο εύχρηστες και δημοφιλείς μεθόδους επίλυσης προβλημάτων.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Αριθμητική Ανάλυση, Διαφορικές εξισώσεις, Nystrom, Runge Kutta, Fehlberg, προβλήματα αρχικών τιμών, Hewn

ABSTRACT

A lot of natural phenomena are described mathematically by differential equations ordinary differential equations or partial differential equations. Unfortunately few differential equations are solved analytically for this reason from the beginning of century became efforts in order to found numerical methods for the solution of differential equations. The main part of research in the numerical solution of differential equations began from the creation of first computers.

The numerical solution of differential equations consists of the construction of algorithms and in continuation their realization with computational PC. Big part of research in the applied mathematics constitutes the construction of such algorithms.

The numerical methods of solving ordinary differential equations which solve problems with initial conditions first and second order (all the other problems are reduced in such problems).

Two big categories of methods for the solution of differential equations are the methods of simple step and the Multistep methods. There are several single steps methods. Among them are the methods of Runge Kutta and Runge Kutta Nystrom. The big precision and the facility of application of algorithms Runge-Kutta and Runge Kutta Nystrom classify them in the most useful and popular methods of solving such problems.

KEY WORDS

Numerical Analysis, Differential equations, Nystrom, Runge Kutta, Fehlberg, problems of initial prices, Hewn

ΠΕΡΙΕΧΟΜΕΝΑ

Ευχαριστώ πολύ την επιβλέπουσα καθηγήτρια μου κ. Γουσίδου-Κουτίτα Μαρία που μου έδωσε αρχικά το κίνητρο για την παρούσα εργασία και για την πολύτιμη βοήθεια της στη συγγραφή και παρουσίαση της!

Αφιερώνεται, στους γονείς μου
και στον αδερφό μου.

ΚΕΦΑΛΑΙΟ 1

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗ ΛΥΣΗ ΚΑΝΟΝΙΚΩΝ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

1.1 Η ΔΙΑΦΟΡΙΚΗ ΕΞΙΣΩΣΗ ΚΑΙ Ο ΤΡΟΠΟΣ ΕΠΙΛΥΣΗΣ ΤΗΣ

Ορισμός 1.1.1:

Μια διαφορική εξίσωση είναι μία εξίσωση όπου εμφανίζεται μία άγνωστη συνάρτηση μαζί με τις αντίστοιχες κ-τάξης παραγώγους της. Η λύση αυτής της διαφορικής εξίσωσης είναι μια συνάρτηση που επαληθεύει την εξίσωση μας.

Πρόταση 1.1.1:

Μια γραμμική διαφορική εξίσωση πρώτης τάξης είναι μια διαφορική εξίσωση της μορφής $y' + p(x)y = q(x)$, όπου p και q είναι γνωστές συναρτήσεις της ανεξάρτητης μεταβλητής χ που είναι συνεχείς σε ένα διάστημα της πραγματικής ευθείας.
Οι λύσεις της γραμμικής διαφορικής εξίσωσης πρώτης τάξης δίνονται από τον τύπο

$$y = e^{-\int p(x)\,dx}\left[c + \int q(x)e^{\int p(x)\,dx}\,dx\right], \qquad (1.1)$$

όπου c σταθερά.

Απόδειξη:

Πολλαπλασιάζοντας και τα δύο μέλη της $y' + p(x)y = q(x)$ με $e^{\int p(x)\,dx}$,

παίρνουμε $y'e^{\int p(x)\,dx} + p(x)ye^{\int p(x)\,dx} = q(x)e^{\int p(x)\,dx}$ ή

$\left[ye^{\int p(x)\,dx}\right]' = q(x)e^{\int p(x)\,dx}$.

Έτσι, έχουμε $ye^{\int p(x)\,dx} = c + \int q(x)e^{\int p(x)\,dx}\,dx$, όπου c είναι μια αυθαίρετη σταθερά.

Παράδειγμα 1.1.1:

Η διαφορική εξίσωση $y' - y = e^x$ είναι πρώτης τάξης η οποία είναι της μορφής:

$y' + p(x)y = q(x)$.

Άρα έχουμε $p(x) = -1$ και $q(x) = e^x$.

Έτσι όλες οι λύσεις της περιγράφονται σύμφωνα με τον τύπο 1.1:

$y = e^{-\int 1\,dx}.[c + \int e^x e^{\int -1\,dx}\,dx] = e^x[c + \int e^x\, e^{-x}\,dx] = e^x[c + \int 1\,dx] =$
$= e^x[c + x]$

Επομένως έχουμε $y(x) = ce^x + xe^x$

Παράδειγμα 1.1.2:

Η διαφορική εξίσωση $y' + (tgx)y = sinx$ είναι πρώτης τάξης η οποία είναι της μορφής:

$y' + p(x)y = q(x)$.

Άρα έχουμε $p(x) = tanx$ $και$ $q(x) = sinx$.

Έτσι όλες οι λύσεις της περιγράφονται σύμφωνα με τον τύπο 1.1:

$$y = e^{-\int tanx\,dx}[c + \int sinx\, e^{\int tanx\,dx}\,dx]$$

$$\int tanx\,dx = \int \frac{sinx}{cosx}\,dx = -\int \frac{(cosx)'}{cosx}\,dx = -\log(cosx)$$

$$\int (sinx)e^{-\log(cosx)}dx = \int (sinx)(cosx)^{-1}dx = \int \frac{sinx}{cosx}dx = -\log(cosx)$$

Άρα, $y = e^{\log(cosx)}[c + (-\log(cosx)] = (cosx)[c - \log(cosx)]$

Παρατηρούμε ότι πήραμε θετικές τιμές για το συνημίτονο οπότε δε χρειάζεται να βάλουμε απόλυτο μέσα στον λογάριθμο.

Παράδειγμα 1.1.3:

Η διαφορική εξίσωση $xy' - 2y = -x^2$ $με$ $αρχικές$ $συνθήκες$ $y(1) = 0$ επιλύεται όπως τις διαφορικές εξισώσεις πρώτης τάξης με τη διαφορά ότι έχουμε αρχικές συνθήκες οι οποίες θα μας βοηθήσουν για να εντοπίσουμε την ακριβή τιμή της αυθαίρετης σταθεράς c.

Έτσι έχουμε: Αρχικά διαιρούμε παντού με χ και η δ. ε. παίρνει τη μορφή:

$$y' - \frac{2y}{x} = -x^1$$

$$p(x) = -\frac{2}{x}, q(x) = -x$$

Έτσι, όλες οι λύσεις της είναι

$$y = e^{-\int -\frac{2}{x}dx}[c + \int (-x)e^{\int -\frac{2}{x}dx}dx\ ]$$

$$\int -\frac{2}{x}\,dx = -2\log|x| = -log|x|^2$$

$$\int (-x)e^{-log|x|^2}dx = \int (-x)(x^{-2)}dx = \int -x^{-1}\,dx = -\log|x|$$

Άρα,

$$y = e^{logx^2}[c - \log|x|] = x^2(c - \log|x|)$$

Οπού c αυθαίρετη σταθερά και για x>0 έχουμε:

$$y = e^{logx^2}[c - \log x] = x^2(c - logx)$$

Έχουμε: y(1)=0

Άρα,

$$0 = 1^2(c - log1) \Rightarrow c = 0$$

Άρα,

$$y = x^2(0 - logx) = -x^2 logx$$

ΧΡΗΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ MATLAB ΓΙΑ ΤΗΝ ΕΠΑΛΗΘΕΥΣΗ ΤΩΝ ΛΥΣΕΩΝ ΤΩΝ ΠΑΡΑΔΕΙΓΜΑΤΩΝ

Παράδειγμα 1.1.1:

```
>> dsolve('Dx=e^t+x')

 ans =

 1/(log(e)-1)*e^t+exp(t)*C1
```

Παράδειγμα 1.1.2:

```
dsolve('Dy=-tan(x)*y+sin(x)')

ans =

 cos(x)*exp((-tan(x)*t*cos(x)+sin(x)*t)/cos(x))+exp(-tan(x)*t)*C1
```

Παράδειγμα 1.1.3:

```
dsolve('Dy=-2*(y/x)-(1/x)' , 'y(1)=0')

ans =

-1/2+1/2*exp(-2/x*t)/exp(-2/x)
```

1.2 ΥΠΑΡΞΗ ΚΑΙ ΜΟΝΑΔΙΚΟΤΗΤΑ ΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΟΣ ΑΡΧΙΚΗΣ ΤΙΜΗΣ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

Ορισμός 1.2.1

Μια συνάρτηση f(x,y) ικανοποιεί μία συνθήκη Lipschitz ως προς τη μεταβλητή y πάνω στο σύνολο $D\subseteq R^2$ αν υπάρχει μία σταθερή L>0 που να ικανοποιεί τη σχέση

$|f(x,y_1)-f(x,y_2)|\leq L|y_1-y_2|$ $\forall\ (x,y_1),(x,y_2)\in D$

Η σταθερή L καλείται σταθερή Lipschitz για την f.

Ορισμός 1.2.2

Ένα σύνολο $D\subseteq R^2$ καλείται κυρτό αν για κάθε $(x_1,y_1),(x_2,y_2)\in D$ το σημείο $((1-\lambda)\chi_1+\lambda\chi_2,(1-\lambda)y_1+\lambda y_2)$ ανήκει επίσης στο D $\forall\ \lambda$ με $0\leq \lambda \leq 1$.

Θεώρημα 1.2.1

Αν f(x,y) ορίζεται πάνω σε ένα κυρτό σύνολο $D\subseteq R^2$ και αν υπάρχει σταθερή L>0 τέτοια ώστε

$$\left|\frac{\partial f}{\partial y}(x,y)\right| \leq L\ \forall\ (x,y)\in D$$

τότε η f ικανοποιεί μία συνθήκη Lipschitz στο D ως προς την μεταβλητή y με σταθερή Lipschitz L.

Θεώρημα 1.2.2

Αν $D=\{(x,y)/\ a\leq x\leq b, -\infty < y < \infty\}$ και f(x,y) συνεχής στο D και ικανοποιεί μία συνθήκη Lipschitz στο D ως προς y, τότε το πρόβλημα αρχικής τιμής

$$y'=f(x,y),\quad a\leq x\leq b,\ y(a)=y_0$$

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

Ορισμός 1.2.3

Το αρχικό πρόβλημα $y'=f(x,y)$, $a\leq x\leq b$, $y(a)=y_0$ λέγεται ότι είναι καλά τοποθετημένο αν:

I. Υπάρχει μία μοναδική λύση y(x) του προβλήματος
II. Υπάρχει ε>0 με την ιδιότητα ότι υπάρχει μία μοναδική λύση, u(x) του προβλήματος

u'=f(x,y)+δ(x), $a \leq x \leq b$, u(a)=$y_0+\varepsilon_0$

για κάθε $|\varepsilon_0|<\varepsilon$ και $|\delta(x)|<\varepsilon$ $\forall x \in [a,b]$

III. Υπάρχει κ>0 έτσι ώστε
$|u(x)-y(x)|<\kappa\varepsilon$ $\forall x \in [a,b]$

Το πρόβλημα της δεύτερης παρατήρησης καλείται διαταραγμένο πρόβλημα του αρχικού προβλήματος του θεωρήματος 2.

Θεώρημα 1.2.3

Αν D={(x,y)/ $a \leq x \leq b,\ c \leq y \leq d$} το πρόβλημα αρχικής τιμής του θεωρήματος 2

$$y'=f(x,y),\quad a \leq x \leq b,\ y(a)=y_0$$

είναι καλά τοποθετημένο αρκεί η f να είναι συνεχής και να ικανοποιεί μία συνθήκη Lipschitz ως προς y στο D.

ΕΥΣΤΑΘΕΙΑ

Ορισμός 1.2.4

Μια μέθοδος εξίσωσης διαφορών με τοπικό σφάλμα αποκοπής ε_i στο i-βήμα της μεθόδου λέγεται ότι είναι συνεπής με τη διαφορική εξίσωση που προσεγγίζει αν

$$\lim_{h\to 0} \max_{1\leq i\leq N} |\varepsilon_i| = 0$$

Ορισμός 1.2.5

Μία μέθοδος εξίσωσης διαφορών λέμε ότι είναι συγκλίνουσα ως προς τη διαφορική εξίσωση που προσεγγίζει αν

$$\lim_{h\to 0} \max_{1\leq i\leq N} |y(x_i) - y_i| = 0,$$

όπου $y(x_i)$ είναι η ακριβής τιμή της λύσης στο x_i και y_i η προσεγγιστική που λαμβάνεται από τη μέθοδο των διαφορών στο i-βήμα.

Ορισμός 1.2.6

Μία μέθοδος εξίσωσης διαφορών που προσεγγίζει μία διαφορική εξίσωση είναι ευσταθής όταν μικρές αλλαγές ή διαταραχές στις αρχικές συνθήκες παράγουν αντίστοιχα μικρές αλλαγές στην ακολουθία των προσεγγίσεων.

Θεώρημα 1.2.4

Αν το πρόβλημα αρχικής τιμής

$$y'=f(x,y),\quad a \leq x \leq b,\ y(a)=y_0$$

προσεγγίζεται από μία μέθοδο διαφορών της μορφής

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

και η φ(x,y,h) ικανοποιεί μία συνθήκη Lipschitz ως προς y στο D

$$D=\{(x,y,h)/\ a\leq x\leq b,\ -\infty<y<\infty, o\leq h\leq h_0\}$$

τότε η μέθοδος είναι ευσταθής.

Μία εξίσωση, που περιλαμβάνει μία σχέση μεταξύ μίας άγνωστης συνάρτησης y και μίας ή περισσοτέρων παραγώγων της, καλείται διαφορική εξίσωση. Μια κανονική διαφορική εξίσωση τάξης n έχει τη μορφή:

$$\frac{d^n y}{dx^n}=f(x,y,\frac{dy}{dx},\frac{d^2y}{dx^2},\dots,\frac{d^{n-1}y}{dx^{n-1}}) \qquad (1)$$

όπου $\frac{d^n y}{dx^n}=y^{(n)}$ είναι η n-οστή παράγωγος της y.

Η λύση της διαφορικής εξίσωσης (1) είναι μία συνάρτηση y(x) που ικανοποιεί την (1), δηλαδή:

$$\frac{d^n y(x)}{dx^n}=f(x,y(x),\frac{dy(x)}{dx},\frac{d^2y(x)}{dx^2},\dots,\frac{d^{n-1}y(x)}{dx^{n-1}}) \qquad (2)$$

και είναι φανερό ότι η συνάρτηση y(x) πρέπει να είναι n-φορές παραγωγίσιμη.

Όταν εκτός από την δ. ε. (1) μας δοθούν και οι αρχικές συνθήκες

$$y(x_0)=y_0,\ y'(x_0)=y'_0\ ,\ y''(x_0)=y''_0\ ,\dots,\ y^{(n-1)}(x_0)=y_0^{(n-1)} \qquad (3)$$

τότε λέμε ότι έχουμε ένα πρόβλημα αρχικών τιμών και σ' αυτό αντιστοιχεί, με κατάλληλες προϋποθέσεις, μία μοναδική λύση.

Όταν οι παραπάνω συνθήκες (3) αναφέρονται σε περισσότερα από ένα σημεία και όχι μόνο στο x_0 τότε οι συνθήκες λέγονται **οριακές συνθήκες** και το πρόβλημα **πρόβλημα οριακών συνθηκών.**

Πολλά προβλήματα των Θετικών Επιστημών μπορούν να καταλήξουν σε διαφορικές εξισώσεις που η εύρεση της λύσης τους αναλυτικά είτε είναι επίπονη και αρκετά δύσκολη είτε η λύση η ίδια αυτή καθ' αυτή είναι αρκετά περίπλοκη ώστε καθιστά την εύρεση των τιμών της λύσης σε ορισμένα σημεία αρκετά δύσκολη. Για το σκοπό αυτό επινοήθηκαν οι αριθμητικές μέθοδοι υπολογισμού της λύσης y(x) μιας διαφορικής εξίσωσης της μορφής (1) με αρχικές συνθήκες (3) ώστε η λύση να είναι μοναδική.

Κάθε δ. ε. της μορφής (1) μπορεί να καταλήξει σε σύστημα n διαφορικών εξισώσεων $1^{ης}$ τάξης. Αυτό πετυχαίνεται με τις αντικαταστάσεις:

$$y' = z_1$$
$$y'' = z_1' = z_2$$
$$y''' = z_2' = z_3$$
$$\dots\dots\dots\dots\dots\dots$$
$$\dots\dots\dots\dots\dots\dots\dots$$
$$y^{(n-1)} = z_{n-2}' = z_{n-1}$$
$$y^{(n)} = z_{n-1}' = f(x, y, z_1, z_2, \dots, z_{n-1})$$

(4)

Το σύστημα (4) είναι ισοδύναμο με την εξίσωση (1). Έτσι, οι αριθμητικές μέθοδοι που θα αναπτυχθούν θα αφορούν τις δ. ε. $1^{ης}$ τάξης της μορφής:

$$y'=f(x,y), \qquad y(x_0)=y_0 \qquad (5)$$

που μπορούν να γενικευτούν και για συστήματα δ. ε. $1^{ης}$ τάξης της μορφής (4).

Υπάρχουν αρκετές καλές μέθοδοι για τον υπολογισμό της λύσης y(x) της δ. ε. (5) σε διαδοχικά ισαπέχοντα σημεία $x_0,x_1,\dots,x_n,\dots$, με βήμα h, που με τη βοήθεια των υπολογιστών πετυχαίνουμε με επιθυμητή ακρίβεια τη λύση της διαφορικής εξίσωσης.

Η αριθμητική λύση του προβλήματος αρχικών τιμών (5) σε κάποιο διάστημα [a,b] συνίσταται στην εύρεση των τιμών της λύσης y(x) στα ισαπέχοντα σημεία $x_i=x_0+ih$,i=1,2,…,n, που οι τιμές αυτές υπολογίζονται από κάποια αριθμητική μέθοδο.

Αν συμβολίσουμε με y_{ip} τις αντίστοιχες τιμές της λύσης στα σημεία x_i, i=0,1,…,n, που παίρνουμε με κάποια αριθμητική μέθοδο και με y_{ia} τις τιμές της ακριβούς λύσης που προκύπτει λύνοντας την (5) αναλυτικά (όπως υποδείξαμε στην πρώτη ενότητα αυτού του κεφαλαίου), τότε λέμε ότι το τοπικό σφάλμα αποκοπής στο διάστημα $[x_{i-1},x_i]$ θα είναι $E_i=|y_{ia}-y_{ip}|$και το ολικό σφάλμα που γίνεται μέσα στο διάστημα [a,b] θα είναι το άθροισμα των παραπάνω σφαλμάτων:

$$E_{ολ}=\sum_{i=1}^{n} |(y_{ia} - y_{ip}| \qquad (6)$$

Οι τύποι της αριθμητικής λύσης διαφορικής εξίσωσης πρώτης τάξης της μορφής (5) χωρίζονται σε δύο κατηγορίες:

I. Σ’ αυτούς που για τον υπολογισμό ενός y_i χρησιμοποιούν μόνον την αμέσως προηγούμενη τιμή, την y_{i-1} για αυτό και λέγονται τύποι απλού βήματος και

II. Σ' αυτούς που για τον υπολογισμό ενός y_i χρησιμοποιούν τις τιμές των $y_{i-1}, y_{i-2}, y_{i-3}, \ldots$ για αυτό και οι τύποι αυτοί καλούνται τύποι πολλαπλού βήματος.

ΚΕΦΑΛΑΙΟ 2

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗ ΛΥΣΗ ΚΑΝΟΝΙΚΩΝ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΠΡΩΤΗΣ ΚΑΙ ΔΕΥΤΕΡΗΣ ΤΑΞΗΣ

2.1 ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΜΕΘΟΔΩΝ ΠΡΩΤΗΣ ΤΑΞΗΣ

2.1.1 Μέθοδος Euler

Αν y(x) είναι η ακριβής λύση της δ. ε. (5) τότε από τον ορισμό της παραγώγου της συνάρτησης y(x) έχουμε:

$$y'(x) = \lim_{h \to 0} \frac{y(x+h) - y(x)}{h} \quad (7)$$

Θεωρώντας το h πολύ μικρό μπορούμε από τον παραπάνω τύπο να πάρουμε προσεγγιστικά:

$$y'(x) \cong \frac{y(x+h) - y(x)}{h}$$

Ή

$$y(x+h) \cong y(x) + hy'(x) \quad (8)$$

Η (8) ξεκινώντας με $x=x_0$ γίνεται:

$$y_1 \cong y_0 + hy_0'$$

στο επόμενο βήμα θα είναι:

$$y_2 \cong y_1 + hy_1'$$

.....

και γενικότερα

$$y_{n+1} \cong y_n + hy_n^{'}, \qquad n = 0,1,2,\ldots, \qquad (9)$$

με $y_n^{'} = f(x_n, y_n)$, όπως προκύπτει από την (5) και $x_n=x_0+nh$.

Η μέθοδος αυτή όπου κάθε τιμή y_{n+1} δίνεται από την προηγούμενη της y_n δια μέσου του τύπου (9) καλείται μέθοδος του Euler. Η μέθοδος αυτή είναι η πιο παλιά για την αριθμητική επίλυση διαφορικών εξισώσεων αλλά είναι και η λιγότερο ακριβής, γι' αυτό αποφεύγεται συνήθως στις πρακτικές εφαρμογές.

Όμως είναι και η μοναδική μέθοδος πρώτης τάξης που γνωρίζουμε και θα ασχοληθούμε σε αυτή την εργασία.

Αν $y_a(x)$ είναι η ακριβής λύση της διαφορικής εξίσωσης (5) και με y_{ip} συμβολίσουμε τις προσεγγιστικές τιμές της λύσης στα σημεία χ_i, i=0,1,2,3,…, τότε το τοπικό σφάλμα αποκοπής θα είναι η διαφορά:

$$E_i = |y_{ia} - y_{ip}|$$

Θεώρημα 2.1.1

Αν η παράγωγος της f(x,y) ως προς y ικανοποιεί την ανίσωση:

$$|f_y(x,y)| < M$$

Στο διάστημα $[x_0, x_n]$, όπου M είναι μια θετική σταθερή και η δεύτερη παράγωγος της ακριβούς λύσης $y_a(x)$ ικανοποιεί την:

$$|y_a^{''}(x)| < L$$

στο διάστημα $[x_0, x_n]$, όπου L επίσης θετική σταθερή, τότε αποδείχνεται ότι το τοπικό σφάλμα αποκοπής ε_i της μεθόδου Euler στο σημείο $x_i=x_0+ih$ είναι μικρότερο ή ίσο από την ποσότητα:

$$|\varepsilon_i| \leq \frac{hL}{2M}(e^{(x_i-x_0)M} - 1)$$

Δηλαδή, το σφάλμα αποκοπής στη μέθοδο του Euler είναι της τάξης O(h) ή c·h, όπου c σταθερή και το σφάλμα αυτό τείνει στο μηδέν όταν $h \to 0$.

Παράδειγμα 2.1.1

Να λυθεί η διαφορική εξίσωση y'=2x-y, y(0)=1, από x=0 μέχρι x=1 με h=0.25, χρησιμοποιώντας την μέθοδο Euler.

Λύση

Θα χρησιμοποιήσουμε τον τύπο (9) και επομένως επειδή έχουμε βήμα h=0.25 και το χ ανήκει στο διάστημα [0,1] θα χρησιμοποιήσουμε τον παραπάνω τύπο για 4 επαναλήψεις.

Τα χ_0 και y_0 προκύπτουν από τις αρχικές τιμές.

Δηλαδή είναι:

$\chi_0=0$

και

$y_0=1$

Οι επόμενες τιμές των 4 επαναλήψεων υπολογίστηκαν στο matlab(υπάρχει ο αντίστοιχος κώδικας στο παράρτημα) και φαίνονται στον παρακάτω πίνακα.

i	x(i)	y(i)
0	0	1
1	0,25	0,75
2	0,5	0,6875
3	0,75	0,765625
4	1	0,94921875

Για να υπολογίσουμε το σφάλμα θα πρέπει αρχικά να υπολογίσουμε την κανονική λύση της διαφορικής μας εξίσωσης. Αυτή εύκολα διαπιστώνεται ότι είναι η ακόλουθη:

$$y(x) = -2 + 2x + 3e^{-x}$$

Τώρα για να παρατηρήσουμε πόσο ικανοποιητική και ακριβής είναι η παραπάνω μέθοδος θα υπολογίσουμε το αντίστοιχο σφάλμα που προκύπτει αφού πρώτα βρούμε τις ακριβείς τιμές.

Αν συμβολίσουμε με g(x) τις ακριβής λύσεις της διαφορικής μας εξίσωσης τότε το αντίστοιχο σφάλμα εμφανίζεται στον παρακάτω πίνακα και το συμβολίζουμε με E.

i	x(i)	y(i)	g(i)	E
0	0	1	1,000000000000000	0
1	0,25	0,75	0,836402349214215	0,086402349
2	0,5	0,6875	0,819591979137900	0,132091979
3	0,75	0,765625	0,917099658223044	0,151474658
4	1	0,94921875	1,103638323514320	0,154419574

Παρατηρούμε ότι έχουμε αρκετά μεγάλες αποκλίσεις και αυτό εξηγείται από το γεγονός ότι έχουμε αρκετά μεγάλο βήμα.

Για τον λόγο αυτό θα ξανά λύσουμε το παραπάνω παράδειγμα με μικρότερο βήμα και συγκεκριμένα h=0.1.

Οι τιμές της προσέγγισης μας δίνονται από τον παρακάτω πίνακα:

i	x(i)	y(i)
0	0	1,000000000000000
1	0,1	0,900000000000000
2	0,2	0,830000000000000
3	0,3	0,787000000000000
4	0,4	0,768300000000000
5	0,5	0,771470000000000
6	0,6	0,794323000000000
7	0,7	0,834890700000000

8	0,8	0,891401630000000
9	0,9	0,962261467000000
10	1	1,046035320300000

Τώρα αν θέσουμε με g(x) την ακριβή λύση της διαφορικής εξίσωσης (δηλαδή $g(x) = -2 + 2x + 3e^{-x}$) και με E το αντίστοιχο σφάλμα όπως περιγράφηκε στο θεώρημα θα έχουμε τον αντίστοιχο πίνακα:

i	x(i)	y(i)	g(i)	E
0	0	1,000000000000000	1	0
1	0,1	0,900000000000000	0,914512254	0,014512254
2	0,2	0,830000000000000	0,856192259	0,026192259
3	0,3	0,787000000000000	0,822454662	0,035454662
4	0,4	0,768300000000000	0,810960138	0,042660138
5	0,5	0,771470000000000	0,819591979	0,048121979
6	0,6	0,794323000000000	0,846434908	0,052111908
7	0,7	0,834890700000000	0,889755911	0,054865211
8	0,8	0,891401630000000	0,947986892	0,056585262
9	0,9	0,962261467000000	1,019708979	0,057447512
10	1	1,046035320300000	1,103638324	0,057603003

Παρατηρούμε ότι τώρα που μικραίναμε το βήμα μας έχουμε καλύτερα αποτελέσματα και αυτό διαπιστώνεται από το γεγονός ότι τα τοπικά σφάλματα είναι μικρότερα από την πρώτη περίπτωση.

Παράδειγμα 2.1.2

Να λυθεί με τη μέθοδο Euler η διαφορική εξίσωση

$$y' = xy + 1, y(0) = 1$$

από x =0 μέχρι x =1 με h=0.25

Λύση

Θα χρησιμοποιήσουμε τον τύπο (9) και επομένως επειδή έχουμε βήμα h=0.25 και το χ ανήκει στο διάστημα [0,1] θα χρησιμοποιήσουμε τον παραπάνω τύπο για 4 επαναλήψεις.
Τα x_0 και y_0 προκύπτουν από τις αρχικές τιμές.

Δηλαδή είναι:

x_0=0

και

y_0=1

Οι επόμενες τιμές των 4 επαναλήψεων υπολογίστηκαν στο matlab(υπάρχει ο αντίστοιχος κώδικας στο παράρτημα) και φαίνονται στον παρακάτω πίνακα.

i	x(i)	y(i)
0	0	1
1	0,25	1,25
2	0,5	1,578125
3	0,75	2,025390625
4	1	2,655151367

Ομοίως όπως και στο προηγούμενο παράδειγμα έχουμε:

i	x(i)	y(i)	g(i)	E
0	0	1	1	0
1	0,25	1,25	1,287017	0,03701743
2	0,5	1,578125	1,676975	0,098849973
3	0,75	2,025390625	2,232585	0,207194422
4	1	2,655151367	3,059407	0,404256038

Παρατηρούμε ότι έχουμε αρκετά μεγάλες αποκλίσεις και αυτό εξηγείται από το γεγονός ότι έχουμε αρκετά μεγάλο βήμα.

Για τον λόγο αυτό θα ξανά λύσουμε το παραπάνω παράδειγμα με μικρότερο βήμα και συγκεκριμένα h=0.1.

Οι τιμές της προσέγγισης μας δίνονται από τον παρακάτω πίνακα:

i	x(i)	y(i)
0	0	1,0000000000000000
1	0,1	1,1000000000000000
2	0,2	1,2110000000000000
3	0,3	1,3352200000000000
4	0,4	1,4752766000000000
5	0,5	1,6342876640000000
6	0,6	1,8160020472000000
7	0,7	2,0249621700320000
8	0,8	2,2667095219342400

9	0,9	2,5480462836889800
10	1	2,8773704492209800

Προσθέτουμε και την ακριβή λύση και το τοπικό σφάλμα και έχουμε τον αντίστοιχο πίνακα:

i	x(i)	y(i)	g(i)	E
0	0	1,0000000000000000	1,0000000000000000	0,0000000000000000
1	0,1	1,1000000000000000	1,1053465218128400	0,0053465218128410
2	0,2	1,2110000000000000	1,2228894624752900	0,0118894624752930
3	0,3	1,3352200000000000	1,3551919637660300	0,0199719637660340
4	0,4	1,4752766000000000	1,5053189529706600	0,0300423529706600
5	0,5	1,6342876640000000	1,6769749725189700	0,0426873085189770
6	0,6	1,8160020472000000	1,8746789919776700	0,0586769447776720
7	0,7	2,0249621700320000	2,1039883184007700	0,0790261483687750
8	0,8	2,2667095219342400	2,3717877696756600	0,1050782477414200
9	0,9	2,5480462836889800	2,6866658536190300	0,1386195699300590
10	1	2,8773704492209800	3,0594074053425700	0,1820369561215890

Παρατηρούμε ότι τώρα που μικραίναμε το βήμα μας έχουμε καλύτερα αποτελέσματα και αυτό διαπιστώνεται από το γεγονός ότι τα τοπικά σφάλματα είναι μικρότερα από την πρώτη περίπτωση. Το ίδιο συμπέρασμα βγάλαμε και στο πρώτο παράδειγμα.

2.1.2 Μέθοδος Runge-Kutta($2^{ης}$ τάξης)

Ο επαναληπτικός τύπος της μεθόδου Runge-Kutta $2^{ης}$ τάξης έχει τη μορφή:

$y_{i+1}=y_i+ak_1+bk_2$

όπου

$k_1=hf(x_i,y_i)$
$k_2=hf(x_i+ch,y_i+dk_1)$

όπου a,b,c,d σταθερές που πρέπει να υπολογιστούν.

Αναπτύσσουμε τώρα την $y(x_{i+1})$ σε σειρά Taylor και έχουμε

$y(x_{i+1})=y(x_i)+hy(x_i)+\frac{h^2}{2!}y''(x_i)+\frac{h^3}{3!}y'''(x_i)+\ldots=$

$=y(x_i)+hf(x_i,y_i)+\frac{h^2}{2}\left(f_x+f_y f\big|_{(x_i,y_i)}\right)+\ldots$

Παρόμοια την $f(x_i+ch,y_i+dk_1)$ την αναπτύσσουμε σε σειρά Taylor που ισχύει για συναρτήσεις δύο μεταβλητών και έχουμε:

$$\frac{k_2}{h}=f(x_i+ch,y_i+dk_1)=f(x_i,y_i)+ch\frac{\partial f}{\partial x}+dk_1\frac{\partial f}{\partial y}+\frac{c^2h^2}{2}\frac{\partial^2 f}{\partial x^2}+\frac{d^2k_1^2}{2}\frac{\partial^2 f}{\partial y^2}+chdk_1\frac{\partial^2 f}{\partial x\partial y}$$

+…

Αν αντικαταστήσουμε την παραπάνω τιμή για το k_2 στην αρχική σχέση που περιγράψαμε στην αρχή της μεθόδου και στην σχέση που περιγράψαμε το κ_1 και εξισώσουμε τους συντελεστές ομοβάθμιων όρων του $h(h,h^2,h^3,\ldots)$ θα προκύψει ένα σύστημα τριών εξισώσεων με τέσσερις αγνώστους (κρατώντας μόνο μέχρι τον όρο με τον συντελεστή h^2)

a+b=1

bc=1/2

bd=1/2

όπου δίνοντας μία αυθαίρετη τιμή στον ένα απ' αυτούς βρίσκουμε τους υπόλοιπους. Η πιο συνηθισμένη μορφή του τύπου Runge-Kutta προκύπτει για a=b=1/2, οπότε c=d=1 και οι τύποι μας παίρνουν τη μορφή:

$y_{i+1}=y_i+1/2(k_1+k_2)$

$k_1=hf(x_i,y_i)$

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

Ο πρώτος τύπος λέγεται βελτιωμένη μέθοδος του Euler ή μέθοδος Heun.

Οι ίδιοι τύποι για b=1, οπότε από το σύστημα προκύπτει a=0, c=d=1/2, γίνονται:

$y_{i+1}=y_i+hf(x_i+h/2,\ y_i+k_1/2)$

και ο επαναληπτικός τύπος λέγεται τροποποιημένη μέθοδος Euler ή μέθοδος του πολυγώνου.

Παράδειγμα 2.1.3

Να λυθεί η διαφορική εξίσωση y'=2x-y, y(0)=1, από x=0 μέχρι x=1 με h=0.25, χρησιμοποιώντας την μέθοδο Runge-Kutta($2^{ης}$ τάξης).

Λύση

Θα χρησιμοποιήσουμε τον τύπο που περιγράψαμε πιο πάνω και επομένως επειδή έχουμε βήμα h=0.25 και το χ ανήκει στο διάστημα [0,1] θα χρησιμοποιήσουμε τον παραπάνω τύπο για 4 επαναλήψεις.

Τα x_0 και y_0 προκύπτουν από τις αρχικές τιμές.

Δηλαδή είναι:

$x_0=0$

και

$y_0=1$

Οι επόμενες τιμές των 4 επαναλήψεων υπολογίστηκαν στο matlab(υπάρχει ο αντίστοιχος κώδικας στο παράρτημα) και φαίνονται στον παρακάτω πίνακα.

i	x(i)	y(i)
0	0	1,0000000000000000
1	0,25	0,8437500000000000
2	0,5	0,8310546875000000
3	0,75	0,9305114746093750
4	1	1,1175870895385700

Ομοίως όπως και στο προηγούμενο παράδειγμα έχουμε:

i	x(i)	y(i)	g(i)	E
0	0	1,0000000000000000	1,000000000000000	0,000000000000000
1	0,25	0,8437500000000000	0,836402349214215	0,007347650785785

2	0,5	0,8310546875000000	0,819591979137900	0,011462708362100
3	0,75	0,9305114746093750	0,917099658223044	0,013411816386331
4	1	1,1175870895385700	1,103638323514320	0,013948766024247

Παρατηρούμε ότι έχουμε αρκετά μεγάλες αποκλίσεις και αυτό εξηγείται από το γεγονός ότι έχουμε αρκετά μεγάλο βήμα.

Για τον λόγο αυτό θα ξανά λύσουμε το παραπάνω παράδειγμα με μικρότερο βήμα και συγκεκριμένα h=0.1.

Οι τιμές της προσέγγισης μας δίνονται από τον παρακάτω πίνακα:

i	x(i)	y(i)
0	0	1,00000000000000000
1	0,1	0,91500000000000000
2	0,2	0,85707500000000000
3	0,3	0,82365287500000000
4	0,4	0,81240585187500000
5	0,5	0,82122729594687500
6	0,6	0,84821070283192200
7	0,7	0,89163068606288900
8	0,8	0,94992577088691500
9	0,9	1,02168282265265000
10	1	1,10562295450065000

Τώρα αν θέσουμε με g(x) την ακριβή λύση της διαφορικής εξίσωσης (δηλαδή

$g(x) = -2 + 2x + 3e^{-x}$) και με E το αντίστοιχο σφάλμα όπως περιγράφηκε στο θεώρημα θα έχουμε τον αντίστοιχο πίνακα:

i	x(i)	y(i)	g(i)	E
0	0	1,0000000000000000	1,0000000000000000	0,0000000000000000
1	0,1	0,9150000000000000	0,9145122541078790	0,0004880000000000
2	0,2	0,8570750000000000	0,8561922592339450	0,0008830000000000
3	0,3	0,8236528750000000	0,8224546620451530	0,0011982129548460
4	0,4	0,8124058518750000	0,8109601381069180	0,0014457137680820
5	0,5	0,8212272959468750	0,8195919791379000	0,0016353168089750
6	0,6	0,8482107028319220	0,8464349082820790	0,0017757945498420
7	0,7	0,8916306860628890	0,8897559113742290	0,0018747746886610
8	0,8	0,9499257708869150	0,9479868923516650	0,0019388785352500
9	0,9	1,0216828226526500	1,0197089792217900	0,0019738434308610
10	1	1,1056229545006500	1,1036383235143200	0,0019846309863290

Παρατηρούμε ότι τώρα που μικραίναμε το βήμα μας έχουμε καλύτερα αποτελέσματα και αυτό διαπιστώνεται από το γεγονός ότι τα τοπικά σφάλματα είναι μικρότερα από την πρώτη περίπτωση.

Παράδειγμα 2.1.4

Να λυθεί με τη μέθοδο Runge-Kutta($2^{ης}$ τάξης) η διαφορική εξίσωση

$y' = xy + 1, y(0) = 1$

από x=0 μέχρι x=1 με h=0.25

Λύση

Θα χρησιμοποιήσουμε τον τύπο που περιγράψαμε στη μέθοδο και επομένως επειδή έχουμε βήμα h=0.25 και το χ ανήκει στο διάστημα [0,1] θα χρησιμοποιήσουμε τον παραπάνω τύπο για 4 επαναλήψεις.

Τα x_0 και y_0 προκύπτουν από τις αρχικές τιμές.

Δηλαδή είναι:

x_0=0

και

y_0=1

Οι επόμενες τιμές των 4 επαναλήψεων υπολογίστηκαν στο matlab(υπάρχει ο αντίστοιχος κώδικας στο παράρτημα) και φαίνονται στον παρακάτω πίνακα.

i	x(i)	y(i)
0	0	1,00000000000000
1	1	1,28515625000000
2	2	1,67112350463867
3	3	2,21808736771345

4	4	3,02612584924645

Ομοίως όπως και στο προηγούμενο παράδειγμα έχουμε:

i	x(i)	y(i)	g(i)	E
0	0	1,000000000000000	1,00000000000000000	0,00000000000000000
1	1	1,285156250000000	1,28701743034606000	0,00186118034606900
2	2	1,671123504638670	1,67697497251897000	0,00585146788030500
3	3	2,218087367713450	2,23258504668044000	0,01449767896699800
4	4	3,026125849246450	3,05940740534257000	0,03328155609612200

Παρατηρούμε ότι έχουμε αρκετά μεγάλες αποκλίσεις και αυτό εξηγείται από το γεγονός ότι έχουμε αρκετά μεγάλο βήμα.

Για τον λόγο αυτό θα ξανά λύσουμε το παραπάνω παράδειγμα με μικρότερο βήμα και συγκεκριμένα h=0.1.

Οι τιμές της προσέγγισης μας δίνονται από τον παρακάτω πίνακα:

i	x(i)	y(i)
0	0	1,0000000000000000
1	0,1	1,1052500000000000
2	0,2	1,2226616437500000
3	0,3	1,3547838502546800
4	0,4	1,5046625465349800

5	0,5	1,6759765574209400
6	0,6	1,8732097358455400
7	0,7	2,1018711276604000
8	0,8	2,3687788739450400
9	0,9	2,6824289264017800
10	1	3,0534770580703200

Δίνουμε και την ακριβή λύση και το τοπικό σφάλμα και έχουμε τον αντίστοιχο πίνακα:

i	x(i)	y(i)	g(i)	E
0	0	1,0000000000000000	1,00000000000000000	0,00000000000000000
1	0,1	1,1052500000000000	1,10534652181284000	0,00096500000000000
2	0,2	1,2226616437500000	1,22288946247529000	0,00022800000000000
3	0,3	1,3547838502546800	1,35519196376603000	0,00040800000000000
4	0,4	1,5046625465349800	1,50531895297066000	0,00065600000000000
5	0,5	1,6759765574209400	1,67697497251897000	0,00099800000000000
6	0,6	1,8732097358455400	1,87467899197767000	0,00146925613212600
7	0,7	2,1018711276604000		
8	0,8	2,3687788739450400		
9	0,9	2,6824289264017800		
10	1	3,0534770580703200		

2,10398831840077000	0,00211719074036900
2,37178776967566000	0,00300889573061500
2,68666585361903000	0,00423692721725100
3,05940740534257000	0,00593034727225200

Παρατηρούμε ότι τώρα που μικρύναμε το βήμα μας έχουμε καλύτερα αποτελέσματα και αυτό διαπιστώνεται από το γεγονός ότι τα τοπικά σφάλματα είναι μικρότερα από την πρώτη περίπτωση. Το ίδιο συμπέρασμα βγάλαμε και στο πρώτο παράδειγμα.

ΚΕΦΑΛΑΙΟ 3

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗ ΛΥΣΗ ΚΑΝΟΝΙΚΩΝ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΤΡΙΤΗΣ ΚΑΙ ΤΕΤΑΡΤΗΣ ΤΑΞΗΣ

3.1 ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΜΕΘΟΔΩΝ ΤΡΙΤΗΣ ΤΑΞΗΣ

Εδώ έχουμε τους παρακάτω τύπους που θα περιγράψουμε αναλυτικά. Υπάρχουν δύο ελεύθερες παράμετροι c_2 και c_3 από τους 8 αγνώστους με 6 εξισώσεις και το τοπικό σφάλμα αποκοπής δίνεται από:

$$T(x,y) = \left(\frac{h^4}{4!}\right)\{[1-4(c_2^3w_2 + c_3^3w_3)]D^3f + (1-12c_2^2a_{32}w_3)f_yD^2f$$

$$+(3-24c_2c_3w_3)DfDf_y + f_y^2Df\}$$

3.1.1 Ο Κλασσικός Τύπος Runge-Kutta

Γνωρίζουμε ότι ισχύει από τους τύπους Runge-Kutta

$$k_i = hf(x_n + c_ih, y_n + \sum_{j=1}^{i-1} a_{ij}k_j)$$

$$y_{n+1} = y_n + \sum_{i=1}^{3} w_ik_i$$

Επομένως για i=1,2,3 οι παραπάνω τύποι θα γίνουν

$$k_1 = hf(x_n + c_1 h, y_n + \sum_{j=1}^{1-1} a_{1j} k_j) \Rightarrow$$

$$k_1 = hf(x_n + c_1 h, y_n) \qquad (1)$$

$$k_2 = hf(x_n + c_2 h, y_n + \sum_{j=1}^{2-1} a_{2j} k_j) \Rightarrow$$

$$k_2 = hf(x_n + c_2 h, y_n + a_{21} k_1) \qquad (2)$$

$$k_3 = hf(x_n + c_3 h, y_n + \sum_{j=1}^{3-1} a_{3j} k_j) \Rightarrow$$

$$k_3 = hf(x_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \qquad (3)$$

Και για c_1=0, c_2=1/2, c_3=1, w_1=3/8, w_2=2/3, w_3=1/6, a_{21}=1/2, a_{31}=-1 και a_{32}=2 έχουμε ότι:

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right)$$

$$k_3 = hf(x_n + h, y_n - k_1 + 2k_2)$$

Επομένως ο γενικός μας τύπος παίρνει τη μορφή:

$$y_{n+1} = y_n + \sum_{i=1}^{3} w_i k_i \Rightarrow$$

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 \Rightarrow$$

$$y_{n+1} = y_n + \frac{3}{8}k_1 + \frac{2}{3}k_2 + \frac{1}{6}k_3$$

Παράδειγμα 3.1.1

Να επιλυθεί η διαφορική εξίσωση $y'=-xy^2$ όταν γνωρίζουμε ότι y(0)=1 με h=0.25 και h=0.1.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα h=0.25 και θα συνεχίσουμε με βήμα h=0.1 και θα παραστήσουμε όπως στο προηγούμενο κεφάλαιο τα αποτελέσματα σε πίνακες.

Υπολογίζουμε τις μεθόδους που περιγράφηκαν στην πρώτη ενότητα την ακριβή λύση της διαφορικής μας εξίσωσης η οποία είναι

$$g(x) = \frac{2}{x^2 + 2}$$

Το πρόγραμμα που υπολογίζει τις 5 προσεγγίσεις της διαφορικής μας εξίσωσης παρατίθεται στο παράρτημα και τα αποτελέσματα του παρουσιάζονται στον παρακάτω πίνακα. h=0.25

i	x(i)	y(i)	g(i)	E
1	1,0000000000000000	1,0000000000000000	0,6666666666666670	0,3333333333333330
2	1,2500000000000000	0,9700113932291670	0,5614035087719300	0,4086078844572370
3	1,5000000000000000	0,8771486219115290	0,4705882352941180	0,4065603866174110
4	1,7500000000000000	0,7515050826215790	0,3950617283950620	0,3564433542265170
5	2,0000000000000000	0,6233554684445290	0,3333333333333330	0,2900221351111960

Αν αλλάξουμε τώρα το βήμα μας από 0.25 σε 0.1 προκύπτει ο επόμενος πίνακας.
h=0.1

i	x(i)	y(i)
1,0000000000000000	1,0000000000000000	1,0000000000000000
2,0000000000000000	1,1000000000000000	0,9950331666666670
3,0000000000000000	1,2000000000000000	0,9783455166621170
4,0000000000000000	1,3000000000000000	0,9510069226585940
5,0000000000000000	1,4000000000000000	0,9147257180144570
6,0000000000000000	1,5000000000000000	0,8715939733097660
7,0000000000000000	1,6000000000000000	0,8238133186989110
8,0000000000000000	1,7000000000000000	0,7734608316822240
9,0000000000000000	1,8000000000000000	0,7223294816931410
10,0000000000000000	1,9000000000000000	0,6718495190979850
11,0000000000000000	2,0000000000000000	0,6230770788952500

g(i)	E
0,6666666666666670	0,3333333333333330
0,6230529595015580	0,3719802071651090
0,5813953488372090	0,3969501678249080
0,5420054200542000	0,4090015026043940
0,5050505050505050	0,4096752129639520
0,4705882352941180	0,4010057380156480
0,4385964912280700	0,3852168274708410
0,4089979550102250	0,3644628766719990
0,3816793893129770	0,3406500923801640
0,3565062388591800	0,3153432802388050
0,3333333333333330	0,2897437455619170

Παρατηρούμε ότι όταν μειώσουμε το βήμα έχουμε καλύτερα αποτελέσματα.

3.1.2 Η μέθοδος Nystrom

Σύμφωνα με τους τύπους που περιγράψαμε στην αρχή του κεφαλαίου και για $c_1=0$, $c_2=2/3$, $c_3=2/3$, $w_1=1/4$, $w_2=3/8$, $w_3=3/8$, $a_{21}=2/3$, $a_{31}=0$ και $a_{32}=2/3$ έχουμε ότι:

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{2}{3}h, y_n + \frac{2}{3}k_1\right)$$

$$k_3 = hf(x_n + \frac{2}{3}h, y_n + \frac{2}{3}k_2)$$

Επομένως ο γενικός μας τύπος παίρνει τη μορφή:

$$y_{n+1} = y_n + \sum_{i=1}^{3} w_i k_i \Rightarrow$$

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 \Rightarrow$$

$$y_{n+1} = y_n + \frac{1}{4}k_1 + \frac{3}{8}k_2 + \frac{3}{8}k_3$$

Παράδειγμα 3.1.2

Να επιλυθεί η διαφορική εξίσωση $y'=-xy^2$ όταν γνωρίζουμε ότι $y(0)=1$ με $h=0.25$ και $h=0.1$.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα $h=0.25$ και θα συνεχίσουμε με βήμα $h=0.1$όπως κάναμε και στο προηγούμενο παράδειγμα.

i	x(i)	y(i)	g(i)	E
1	1,00000000000000000	1,00000000000000000	0,66666666666666700	0,33333333333333300
2	1,25000000000000000	0,96960599922839500	0,56140350877193000	0,40820249045646500

3	1,50000000000000000	0,88878847586951600	0,47058823529411800	0,41820024057539800
4	1,75000000000000000	0,78040861669752100	0,39506172839506200	0,38534688830245900
5	2,00000000000000000	0,66656602025201500	0,33333333333333300	0,33323268691868200

Ομοίως για βήμα h=0.1 έχουμε

i	x(i)	y(i)
1,0000000000000000	1,0000000000000000	1,0000000000000000
2,0000000000000000	1,1000000000000000	0,9950221728395060
3,0000000000000000	1,2000000000000000	0,9803872381293220
4,0000000000000000	1,3000000000000000	0,9569314703766760
5,0000000000000000	1,4000000000000000	0,9259190333490610
6,0000000000000000	1,5000000000000000	0,8888820604986000
7,0000000000000000	1,6000000000000000	0,8474511353718690
8,0000000000000000	1,7000000000000000	0,8032066201871010
9,0000000000000000	1,8000000000000000	0,7575694727326880
10,0000000000000000	1,9000000000000000	0,7117370281689460
11,0000000000000000	2,0000000000000000	0,6666590928639060

g(i)	E

0,6666666666666670	0,33333333333333300
0,6230529595015580	0,37196921333794800
0,5813953488372090	0,39899188929211300
0,5420054200542000	0,41492605032247600
0,5050505050505050	0,42086852829855600
0,4705882352941180	0,41829382520448200
0,4385964912280700	0,40885464414379900
0,4089979550102250	0,39420866517687600
0,3816793893129770	0,37589008341971100
0,3565062388591800	0,35523078930976600
0,3333333333333330	0,33332575953057300

Παρατηρούμε ότι ελαφρώς βελτιώνονται οι προσεγγίσεις μας με την αλλαγή του βήματος.

3.1.3 Η μέθοδος Hewn

Σύμφωνα με τους τύπους που περιγράψαμε στην αρχή του κεφαλαίου και για $c_1=0$, $c_2=1/3$, $c_3=1/3$, $w_1=1/4$, $w_2=0$, $w_3=3/4$, $a_{21}=1/3$, $a_{31}=0$ και $a_{32}=2/3$ έχουμε ότι:

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{1}{3}h, y_n + \frac{1}{3}k_1\right)$$

$$k_3 = hf(x_n + \frac{1}{3}h, y_n + \frac{2}{3}k_2)$$

Επομένως ο γενικός μας τύπος παίρνει τη μορφή:

$$y_{n+1} = y_n + \sum_{i=1}^{3} w_i k_i \Rightarrow$$

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 \Rightarrow$$

$$y_{n+1} = y_n + \frac{1}{4}k_1 + \frac{3}{4}k_3$$

Παράδειγμα 3.1.3

Να επιλυθεί η διαφορική εξίσωση $y'=-xy^2$ όταν γνωρίζουμε ότι y(0)=1 με h=0.25 και h=0.1.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα h=0.25 και θα συνεχίσουμε με βήμα h=0.1όπως κάναμε και στο προηγούμενο παράδειγμα.

i	x(i)	y(i)	g(i)	E
1	1,00000000000000000	1,00000000000000000	0,66666666666666700	0,333333333333330
2	1,25000000000000000	0,98480601369598800	0,56140350877193000	0,423402504924058
3	1,50000000000000000	0,91523335198219500	0,47058823529411800	0,444645116688077
4	1,75000000000000000	0,81190234260500900	0,39506172839506200	0,416840614209947
5	2,00000000000000000	0,69787316394222700	0,33333333333333300	0,364539830608894

Ομοίως για βήμα h=0.1 παρατηρούμε

i	x(i)	y(i)
1,0000000000000000	1,0000000000000000	1,0000000000000000
2,0000000000000000	1,1000000000000000	0,9975110987654320

3,0000000000000000	1,2000000000000000	0,9852477541153300
4,0000000000000000	1,3000000000000000	0,9639167071764880
5,0000000000000000	1,4000000000000000	0,9346855826373710
6,0000000000000000	1,5000000000000000	0,8990316832050910
7,0000000000000000	1,6000000000000000	0,8585724783185570
8,0000000000000000	1,7000000000000000	0,8149100492110770
9,0000000000000000	1,8000000000000000	0,7695113167849700
10,0000000000000000	1,9000000000000000	0,7236326807502240
11,0000000000000000	2,0000000000000000	0,6782866264532730

g(i)	E
0,6666666666666670	0,333333333333333000
0,6230529595015580	0,374458139263874000
0,5813953488372090	0,403852405278121000
0,5420054200542000	0,421911287122288000
0,5050505050505050	0,429635077586866000
0,4705882352941180	0,428443447910973000
0,4385964912280700	0,419975987090487000
0,4089979550102250	0,405912094200852000
0,3816793893129770	0,387831927471993000
0,3565062388591800	0,367126441891044000
0,3333333333333330	0,344953293119940000

Η παρατήρηση των προηγουμένων παραδειγμάτων επιβεβαιώνεται και εδώ.

3.2 ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΜΕΘΟΔΩΝ ΤΕΤΑΡΤΗΣ ΤΑΞΗΣ

Εδώ έχουμε πάλι 2 ελεύθερες παραμέτρους που προκύπτουν από 13 αγνώστους με 11 εξισώσεις, αυτοί είναι c_2 και c_3. Το τοπικό σφάλμα δίνεται από

$$T(x,y) = h^5\{[\frac{1}{120} - \frac{w_2c_2^4 + w_3c_3^4 + w_4c_4^4}{24}]D^4f$$

$$+\left[\frac{1}{20}-\frac{w_3c_2c_3^2a_{32}+w_4c_4^2(c_2a_{42}+c_3a_{43})}{2}\right]D^2f_yDf$$

$$+\left[\frac{1}{30}-\frac{w_3a_{32}c_2^2c_3+w_4c_4(a_{42}c_2^2+a_{43}c_3^2)}{2}\right]D\,f_yD^2f$$

$$+\left[\frac{1}{120}-\frac{w_4a_{43}a_{32}c_2^2}{2}\right]f_y^2D^2f$$

$$+\left[\frac{1}{40}-\frac{w_3a_{32}^2c_2^2+w_4(a_{43}c_3+a_{42}c_2)^2}{2}\right]f_{yy}D^2f$$

$$+\left[\frac{1}{120}-\frac{w_3a_{32}c_2^3+w_4(a_{43}c_3^3+a_{42}c_2^3)}{6}\right]f_yD^3f$$

$$+\left[\frac{7}{120}-w_4a_{43}a_{32}c_2(c_3+c_4)\right]f_yDf_yDf+\frac{1}{120}f_y^3Df\}$$

Είναι φανερό ότι T(x,h) είναι ήδη πολύπλοκο που δεν θα προσπαθήσουμε να το γράψουμε για υψηλότερης τάξης περιπτώσεις. Τύποι ειδικού ενδιαφέροντος είναι:

3.2.1 Ο Κλασσικός Τύπος Runge-Kutta

Γνωρίζουμε ότι ισχύει από τους τύπους Runge-Kutta
$k_i = hf(x_n + c_ih, y_n + \sum_{j=1}^{i-1} a_{ij}k_j)$

Και

$$y_{n+1} = y_n + \sum_{i=1}^{4} w_ik_i$$

Επομένως για i=1,2,3,4 οι παραπάνω τύποι θα γίνουν

$$k_1 = hf(x_n + c_1h, y_n + \sum_{j=1}^{1-1} a_{1j}k_j) \Rightarrow$$

$$k_1 = hf(x_n + c_1h, y_n) \qquad (1)$$

$$k_2 = hf(x_n + c_2 h, y_n + \sum_{j=1}^{2-1} a_{2j} k_j) \Rightarrow$$

$$k_2 = hf(x_n + c_2 h, y_n + a_{21} k_1) \qquad (2)$$

$$k_3 = hf(x_n + c_3 h, y_n + \sum_{j=1}^{3-1} a_{3j} k_j) \Rightarrow$$

$$k_3 = hf(x_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \qquad (3)$$

$$k_4 = hf(x_n + c_4 h, y_n + \sum_{j=1}^{4-1} a_{4j} k_j) \Rightarrow$$

$$k_4 = hf(x_n + c_4 h, y_n + a_{41} k_1 + a_{42} k_2 + a_{43} \kappa_3) \quad (4)$$

Και για $c_1=0$, $c_2=1/2$, $c_3=1/2$,$c_4=1$, $w_1=1/6$, $w_2=1/3$, $w_3=1/3$,$w_4=1/6$, $a_{21}=1/2$, $a_{31}=0$,$a_{32}=1/2$,$a_{41}=0$,$a_{42}=0$ και $a_{43}=1$ έχουμε ότι:

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right)$$

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

$$k_4 = hf(x_n + h, y_n + k_3)$$

Επομένως ο γενικός μας τύπος παίρνει τη μορφή:

$$y_{n+1} = y_n + \sum_{i=1}^{4} w_i k_i \Rightarrow$$

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 \Rightarrow$$

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4$$

Παράδειγμα 3.2.1

Να επιλυθεί η διαφορική εξίσωση $y'=-xy^2$ όταν γνωρίζουμε ότι y(0)=1 με h=0.25 και h=0.1.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα h=0.25 και θα συνεχίσουμε με βήμα h=0.1 και θα παραστήσουμε όπως στη προηγούμενη ενότητα τα αποτελέσματα σε πίνακες.

Υπολογίζουμε με τις μεθόδους που περιγράφηκαν στην ενότητα την ακριβή λύση της διαφορικής μας εξίσωσης η οποία είναι

$$g(x) = \frac{2}{x^2 + 2}$$

Το πρόγραμμα που υπολογίζει τις 5 προσεγγίσεις της διαφορικής μας εξίσωσης παρατίθεται στο παράρτημα και τα αποτελέσματα του παρουσιάζονται στον παρακάτω πίνακα.

i	x(i)	y(i)	g(i)	E
1	1,00000000000000000	1,00000000000000000	0,66666666666666700	0,333333333333333
2	1,25000000000000000	0,96969428177241500	0,56140350877193000	0,408290773000485
3	1,50000000000000000	0,88888044489129600	0,47058823529411800	0,418292209597178
4	1,75000000000000000	0,78047784592716000	0,39506172839506200	0,385416117532098

5	2,00000000000000000	0,66666129775504500	0,33333333333333300	0,333327964421712

Αν αλλάξουμε τώρα το βήμα μας από 0.25 σε 0.1 προκύπτει ο επόμενος πίνακας.

i	x(i)	y(i)
1,0000000000000000	1,0000000000000000	1,0000000000000000
2,0000000000000000	1,1000000000000000	0,9950248651026070
3,0000000000000000	1,2000000000000000	0,9803921161009690
4,0000000000000000	1,3000000000000000	0,9569377150604990
5,0000000000000000	1,4000000000000000	0,9259257966013530
6,0000000000000000	1,5000000000000000	0,8888887236836220
7,0000000000000000	1,6000000000000000	0,8474574443936580
8,0000000000000000	1,7000000000000000	0,8032126736506880
9,0000000000000000	1,8000000000000000	0,7575756064040680
10,0000000000000000	1,9000000000000000	0,7117436646314870
11,0000000000000000	2,0000000000000000	0,6666666130890390

g(i)	E
0,6666666666666670	0,333333333333333000
0,6230529595015580	0,371971905601049000
0,5813953488372090	0,398996767263760000
0,5420054200542000	0,414932295006299000
0,5050505050505050	0,420875291550848000
0,4705882352941180	0,418300488389504000
0,4385964912280700	0,408860953165588000
0,4089979550102250	0,394214718640463000

0,3816793893129770	0,375896217091091000
0,3565062388591800	0,355237425772307000
0,3333333333333330	0,333333279755706000

3.2.2 ΚUTTA ΤΥΠΟΣ

Σύμφωνα με τους τύπους που περιγράψαμε στην αρχή του κεφαλαίου και για $c_1=0$, $c_2=1/3$, $c_3=2/3$,$c_4=1$, $w_1=1/8$, $w_2=3/8$, $w_3=3/8$,$w_4=1/8$, $a_{21}=1/3$, $a_{31}=-1/3$,$a_{32}=1$,$a_{41}=1$,$a_{42}=-1$ και $a_{43}=1$ έχουμε ότι:

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{1}{3}h, y_n + \frac{1}{3}k_1\right)$$

$$k_3 = hf(x_n + \frac{2}{3}h, y_n - \frac{1}{3}k_1 + k_2)$$

$$k_4 = hf(x_n + h, y_n + k_1 - k_2 + k_3)$$

Επομένως ο γενικός μας τύπος παίρνει τη μορφή:

$$y_{n+1} = y_n + \sum_{i=1}^{4} w_i k_i \Rightarrow$$

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 \Rightarrow$$

$$y_{n+1} = y_n + \frac{1}{8}k_1 + \frac{3}{8}k_2 + \frac{3}{8}k_3 + \frac{1}{8}k_4$$

Παράδειγμα 3.2.2

Να επιλυθεί η διαφορική εξίσωση $y'=-xy^2$ όταν γνωρίζουμε ότι $y(0)=1$ με $h=0.25$ και $h=0.1$.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα $h=0.25$ και θα συνεχίσουμε με βήμα $h=0.1$όπως κάναμε και στο προηγούμενο παράδειγμα.

i	x(i)	y(i)	g(i)	E
1	1,00000000000000000	1,00000000000000000	0,66666666666666700	0,33333333333333300
2	1,25000000000000000	0,96969008200197600	0,56140350877193000	0,40828657323004600
3	1,50000000000000000	0,88886692881934200	0,47058823529411800	0,41827869352522400
4	1,75000000000000000	0,78045784438386900	0,39506172839506200	0,38539611598880700
5	2,00000000000000000	0,66664017608709600	0,33333333333333300	0,33330684275376300

Ομοίως για βήμα h=0.1 έχουμε

i	x(i)	y(i)
1,0000000000000000	1,0000000000000000	1,0000000000000000
2,0000000000000000	1,1000000000000000	0,9950248477747000
3,0000000000000000	1,2000000000000000	0,9803920492233100
4,0000000000000000	1,3000000000000000	0,9569375752288080
5,0000000000000000	1,4000000000000000	0,9259255732175220
6,0000000000000000	1,5000000000000000	0,8888884191546800
7,0000000000000000	1,6000000000000000	0,8474570711665200

8,0000000000000000	1,7000000000000000	0,8032122497621280
9,0000000000000000	1,8000000000000000	0,7575751512448190
10,0000000000000000	1,9000000000000000	0,7117431959363590
11,0000000000000000	2,0000000000000000	0,6666661453699930

g(i)	E
0,6666666666666670	0,333333333333333000
0,6230529595015580	0,371971888273142000
0,5813953488372090	0,398996700386101000
0,5420054200542000	0,414932155174608000
0,5050505050505050	0,420875068167017000
0,4705882352941180	0,418300183860562000
0,4385964912280700	0,408860579938450000
0,4089979550102250	0,394214294751903000
0,3816793893129770	0,375895761931842000
0,3565062388591800	0,355236957077179000
0,3333333333333330	0,333332812036660000

ΚΕΦΑΛΑΙΟ 4

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗ ΛΥΣΗ ΚΑΝΟΝΙΚΩΝ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΑΝΩΤΕΡΕΩΝ ΤΑΞΕΩΝ

4.1 ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΜΕΘΟΔΩΝ ΑΝΩΤΕΡΗΣ ΤΑΞΗΣ

Καθώς αυξάνεται η τάξη p της δ. ε $p\geq 5$ η πολυπλοκότητα των αλγεβρικών εξισώσεων και η ελεύθερη επιλογή των παραμέτρων οδηγεί σε ένα ευρύ σύνολο τύπων Runge-Kutta. Για p=5, υπάρχουν 16 εξισώσεις και 21 άγνωστοι και έτσι 5 παράμετροι λαμβάνονται αυθαίρετα. Εδώ θα δώσουμε μερικούς τύπους που

προκύπτουν για διάφορες τιμές των αυθαίρετων σταθερών και θα δούμε ποιοι από αυτούς έχουν χαρακτηριστικά ιδιαίτερου ενδιαφέροντος.

Σε αυτή την ενότητα θα ασχοληθούμε με τους πιο διαδεδομένους τύπους και θα περιγράψουμε λεπτομερώς τις σχέσεις τους.

4.1.1 Ο τύπος του Nystrom

Γνωρίζουμε ότι ισχύει από τους τύπους Runge-Kutta
$k_i = \square f(x_n + c_i \square, y_n + \sum_{j=1}^{i-1} a_{ij} k_j)$

Και

$$y_{n+1} = y_n + \sum_{i=1}^{5} w_i k_i$$

Επομένως για i=1,2,3,4,5 οι παραπάνω τύποι θα γίνουν

$$k_1 = hf(x_n + c_1 h, y_n + \sum_{j=1}^{1-1} a_{1j} k_j) \Rightarrow$$

$$k_1 = hf(x_n + c_1 h, y_n) \qquad (1)$$

$$k_2 = hf(x_n + c_2 h, y_n + \sum_{j=1}^{2-1} a_{2j} k_j) \Rightarrow$$

$$k_2 = hf(x_n + c_2 h, y_n + a_{21} k_1) \qquad (2)$$

$$k_3 = hf(x_n + c_3 h, y_n + \sum_{j=1}^{3-1} a_{3j} k_j) \Rightarrow$$

$$k_3 = hf(x_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \qquad (3)$$

$$k_4 = hf(x_n + c_4 h, y_n + \sum_{j=1}^{4-1} a_{4j} k_j) \Rightarrow$$

$$k_4 = hf(x_n + c_4 h, y_n + a_{41} k_1 + a_{42} k_2 + a_{43} \kappa_3) \qquad (4)$$

$$k_5 = hf(x_n + c_5 h, y_n + \sum_{j=1}^{5-1} a_{5j} k_j) \Rightarrow$$

$$k_5 = hf(x_n + c_5 h, y_n + a_{51} k_1 + a_{52} k_2 + a_{53} \kappa_3 + a_{54} \kappa_4) \quad (5)$$

$$k_6 = hf(x_n + c_6 h, y_n + \sum_{j=1}^{6-1} a_{6j} k_j) \Rightarrow$$

$$k_6 = hf(x_n + c_6 h, y_n + a_{61} k_1 + a_{62} k_2 + a_{63} \kappa_3 + a_{64} \kappa_4 + a_{65} \kappa_5) \quad (6)$$

Αν αντικαταστήσουμε τα γνωστά c_i, a_{ij} και w_i προκύπτουν οι εξής τύποι

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + \frac{1}{3} h, y_n + \frac{1}{3} k_1)$$

$$k_3 = hf(x_n + \frac{2}{5} h, y_n + \frac{4}{25} k_1 + \frac{6}{25} k_2)$$

$$k_4 = hf(x_n + h, y_n + \frac{1}{4} k_1 - \frac{12}{4} k_2 + \frac{15}{4} \kappa_3)$$

$$k_5 = hf(x_n + \frac{2}{3} h, y_n + \frac{6}{81} k_1 + \frac{90}{81} k_2 - \frac{50}{81} \kappa_3 + \frac{8}{81} \kappa_4)$$

$$k_6 = hf(x_n + \frac{4}{5} h, y_n + \frac{6}{75} k_1 + \frac{36}{75} k_2 + \frac{10}{75} \kappa_3 + \frac{8}{75} \kappa_4)$$

Και επομένως προκύπτει:

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 + w_5 k_5 + w_6 k_6 \Rightarrow$$

$$y_{n+1} = y_n + \frac{23}{192} k_1 + \frac{195}{192} k_3 - \frac{81}{192} k_4 + \frac{125}{192} k_5$$

Παράδειγμα 4.1.1

Να επιλυθεί η διαφορική εξίσωση y'=-y+x όταν γνωρίζουμε ότι y(0)=1 με h=0.25 και h=0.1.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα h=0.25 και θα συνεχίσουμε με βήμα h=0.1όπως κάναμε και στο προηγούμενο παράδειγμα.

i	x(i)	y(i)	g(i)	E
1	0,00000000000000000	1,00000000000000000	1,00000000000000000	0,0000000000000000
2	0,25000000000000000	0,97392126812126400	0,80760156614281000	0,1663197019784540
3	0,50000000000000000	0,89277859971970300	0,71306131942526700	0,1797172802944360
4	0,75000000000000000	0,77989181057362500	0,69473310548202900	0,0851587050915960
5	1,00000000000000000	0,66044871160543600	0,73575888234288500	0,0753101707374491

Ομοίως για βήμα h=0.1 έχουμε

i	x(i)	y(i)
1,0000000000000000	0,0000000000000000	1,0000000000000000
2,0000000000000000	0,1000000000000000	0,9958606944972650
3,0000000000000000	0,2000000000000000	0,9819090660571790
4,0000000000000000	0,3000000000000000	0,9588763588321920
5,0000000000000000	0,4000000000000000	0,9279707331497740

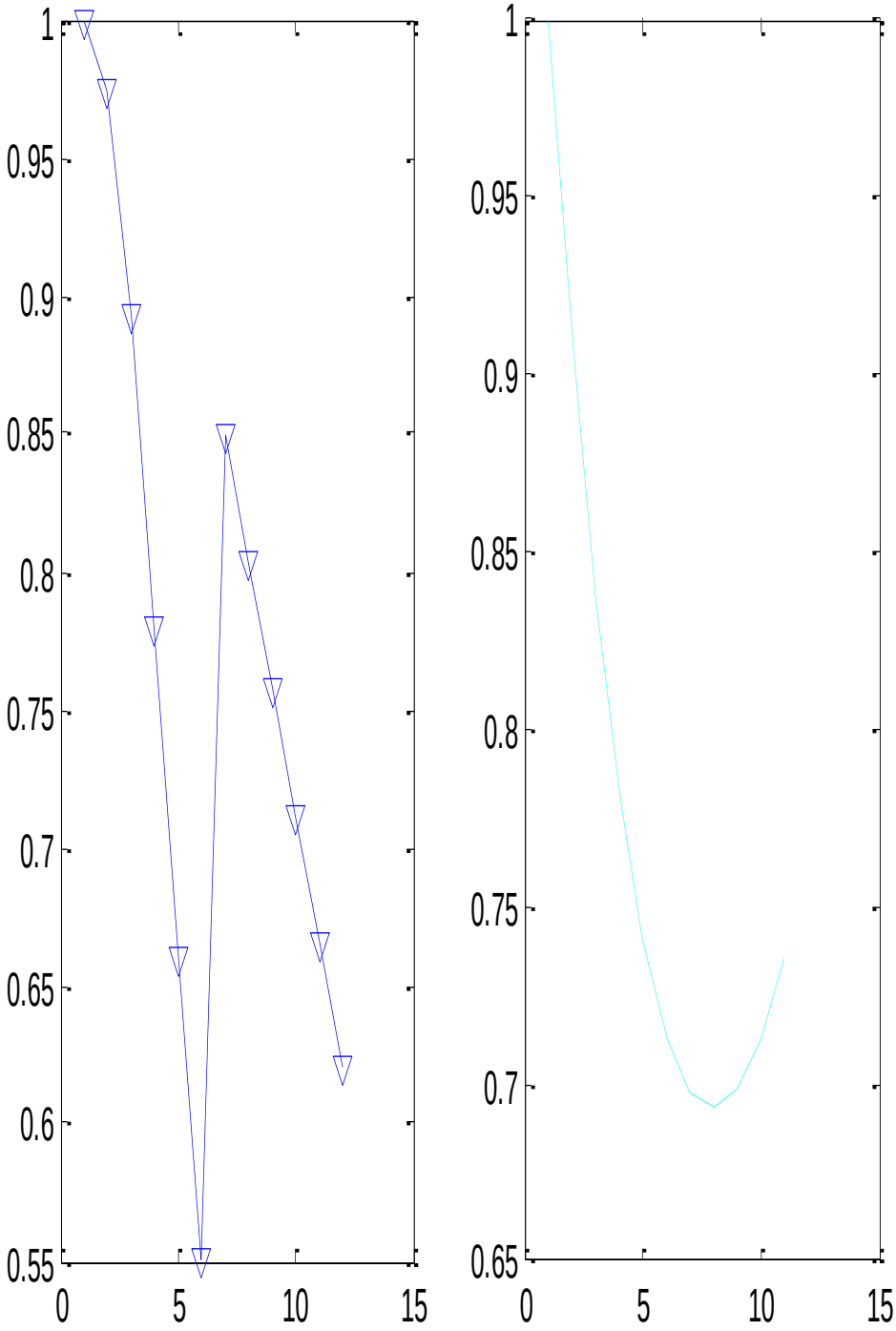
6,0000000000000000	0,5000000000000000	0,8907223217288460
7,0000000000000000	0,6000000000000000	0,8488053251585730
8,0000000000000000	0,7000000000000000	0,8038727235329100
9,0000000000000000	0,8000000000000000	0,7574286978177510
10,0000000000000000	0,9000000000000000	0,7107492502820590
11,0000000000000000	1,0000000000000000	0,6648487321494500

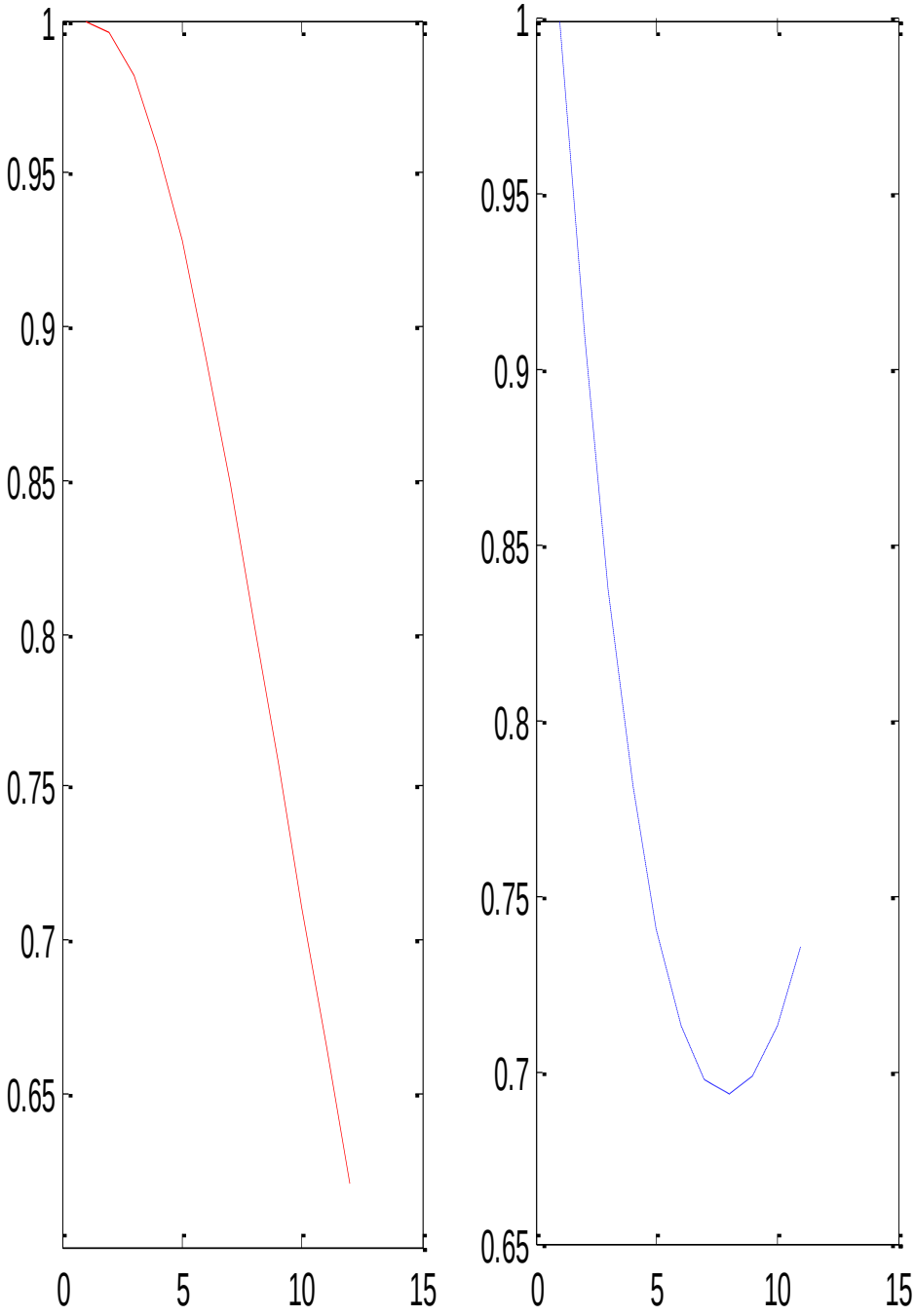
g(i)	E
1,0000000000000000	0,000000000000000000
0,9096748360719190	0,086185858425346000
0,8374615061559640	0,144447559901215000
0,7816364413634360	0,177239917468756000
0,7406400920712790	0,187330641078495000
0,7130613194252670	0,177661002303579000
0,6976232721880530	0,151182052970520000
0,6931706075828190	0,110702115950091000
0,6986579282344430	0,058770769583308100
0,7131393194811980	0,002390069199138980
0,7357588823428850	0,070910150193435100

Θα προσπαθήσουμε να παραστήσουμε τώρα τις γραφικές παραστάσεις της ακριβούς και της προσεγγιστικής λύσης.

Η γραφική παράσταση της προσεγγιστικής λύσης δίνεται από το πρώτο γράφημα ενώ η ακριβής λύση δίνεται από το δεύτερο γράφημα.

Θα παρουσιάσουμε τις γραφικές παραστάσεις στις ανώτερες τάξεις για το λόγο ότι θα είχαμε ένα κουραστικό κείμενο αν το κάναμε σε όλες τις μεθόδους και οι καλύτερες προσεγγίσεις όπως έχουμε περιγράψει πιο πάνω είναι στις ανωτέρω τάξεις.


1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
5
10
15
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15


1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15

4.1.2 Ο τύπος του Fehlberg

Γνωρίζουμε ότι ισχύει από τους τύπους Runge-Kutta

$k_i = hf(x_n + c_i h, y_n + \sum_{j=1}^{i-1} a_{ij} k_j)$

Και

$$y_{n+1} = y_n + \sum_{i=1}^{5} w_i k_i$$

Επομένως για i=1,2,3,4,5 οι παραπάνω τύποι θα γίνουν

$$k_1 = hf(x_n + c_1 h, y_n + \sum_{j=1}^{1-1} a_{1j} k_j) \Rightarrow$$

$$k_1 = hf(x_n + c_1 h, y_n) \qquad (1)$$

$$k_2 = hf(x_n + c_2 h, y_n + \sum_{j=1}^{2-1} a_{2j} k_j) \Rightarrow$$

$$k_2 = hf(x_n + c_2 h, y_n + a_{21} k_1) \qquad (2)$$

$$k_3 = hf(x_n + c_3 h, y_n + \sum_{j=1}^{3-1} a_{3j} k_j) \Rightarrow$$

$$k_3 = hf(x_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) \qquad (3)$$

$$k_4 = hf(x_n + c_4 h, y_n + \sum_{j=1}^{4-1} a_{4j} k_j) \Rightarrow$$

$$k_4 = hf(x_n + c_4 h, y_n + a_{41} k_1 + a_{42} k_2 + a_{43} \kappa_3) \qquad (4)$$

$$k_5 = hf(x_n + c_5 h, y_n + \sum_{j=1}^{5-1} a_{5j} k_j) \Rightarrow$$

$$k_5 = hf(x_n + c_5 h, y_n + a_{51} k_1 + a_{52} k_2 + a_{53} \kappa_3 + a_{54} \kappa_4) \quad (5)$$

$$k_6 = hf(x_n + c_6 h, y_n + \sum_{j=1}^{6-1} a_{6j} k_j) \Rightarrow$$

$$k_6 = hf(x_n + c_6 h, y_n + a_{61} k_1 + a_{62} k_2 + a_{63} \kappa_3 + a_{64} \kappa_4 + a_{65} \kappa_5) \quad (6)$$

Αν αντικαταστήσουμε τα γνωστά c_i, a_{ij} και w_i προκύπτουν οι εξής τύποι

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + \frac{1}{6} h, y_n + \frac{1}{6} k_1)$$

$$k_3 = hf(x_n + \frac{4}{15} h, y_n + \frac{4}{75} k_1 + \frac{16}{75} k_2)$$

$$k_4 = hf(x_n + \frac{2}{3} h, y_n + \frac{5}{6} k_1 - \frac{8}{3} k_2 + \frac{5}{2} \kappa_3)$$

$$k_5 = hf(x_n + \frac{4}{5} h, y_n - \frac{8}{5} k_1 + \frac{144}{25} k_2 - 4\kappa_3 + \frac{16}{25} \kappa_4)$$

$$k_6 = hf(x_n + h, y_n + \frac{361}{320} k_1 - \frac{18}{5} k_2 + \frac{407}{128} \kappa_3 - \frac{11}{80} \kappa_4 + \frac{55}{128} \kappa_5)$$

Και επομένως προκύπτει:

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 + w_5 k_5 + w_6 k_6 \Rightarrow$$

$$y_{n+1} = y_n + \frac{31}{384} k_1 + \frac{1125}{2816} k_3 + \frac{9}{32} k_4 + \frac{125}{768} k_5 + \frac{5}{66} k_6$$

Παράδειγμα 4.1.2

Να επιλυθεί η διαφορική εξίσωση y'=-y+x όταν γνωρίζουμε ότι y(0)=1 με h=0.25 και h=0.1.

Λύση

Αρχικά θα ξεκινήσουμε με βήμα h=0.25 και θα συνεχίσουμε με βήμα h=0.1όπως κάναμε και στο προηγούμενο παράδειγμα.

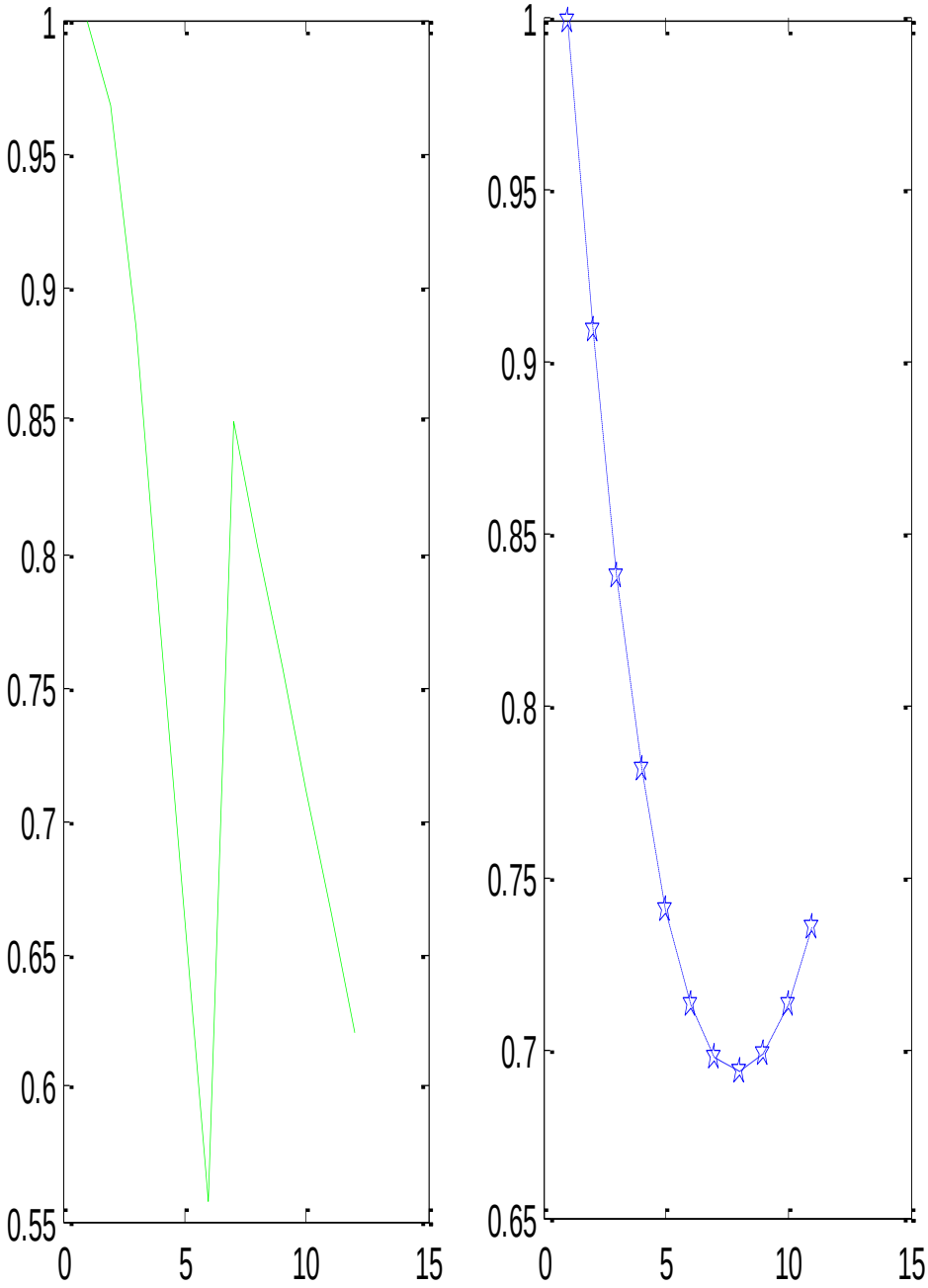
i	x(i)	y(i)	g(i)	E
1	0,00000000000000000	1,00000000000000000	1,00000000000000000	0,00000000000000000
2	0,25000000000000000	0,96748725816639400	0,80760156614281000	0,15988569202358400
3	0,50000000000000000	0,88518054475358300	0,71306131942526700	0,17211922532831600
4	0,75000000000000000	0,77620389181226300	0,69473310548202900	0,08147078633023400
5	1,00000000000000000	0,66250274308956100	0,73575888234288500	0,07325613925332400

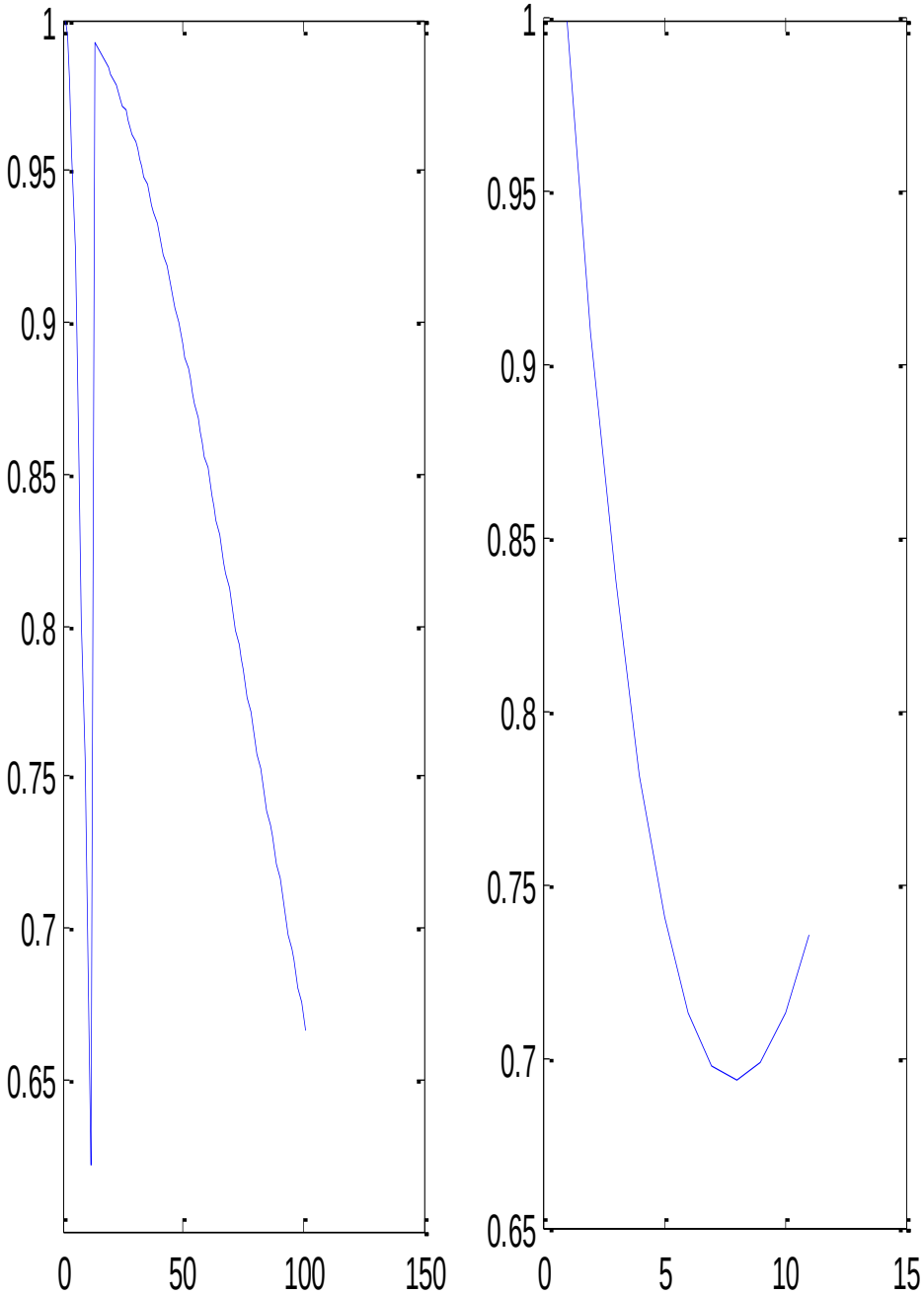
Ομοίως για βήμα h=0.1 έχουμε

i	x(i)	y(i)
1,0000000000000000	0,0000000000000000	1,0000000000000000
2,0000000000000000	0,1000000000000000	0,9946534569351360
3,0000000000000000	0,2000000000000000	0,9796712590369670
4,0000000000000000	0,3000000000000000	0,9559079120251300
5,0000000000000000	0,4000000000000000	0,9246406926095370

6,0000000000000000	0,5000000000000000	0,8874087008231410
7,0000000000000000	0,6000000000000000	0,8458435041295440
8,0000000000000000	0,7000000000000000	0,8015215483094860
9,0000000000000000	0,8000000000000000	0,7558565362836840
10,0000000000000000	0,9000000000000000	0,7100368216745680
11,0000000000000000	1,0000000000000000	0,6650028610289030

g(i)	E
1,0000000000000000	0,000000000000000000
0,9096748360719190	0,084978620863216900
0,8374615061559640	0,142209752881003000
0,7816364413634360	0,174271470661694000
0,7406400920712790	0,184000600538258000
0,7130613194252670	0,174347381397874000
0,6976232721880530	0,148220231941491000
0,6931706075828190	0,108350940726667000
0,6986579282344430	0,057198608049241000
0,7131393194811980	0,003102497806629970
0,7357588823428850	0,070756021313982000


1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
5
10
15
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15


1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
50
100
150
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15

ΣΥΜΠΕΡΑΣΜΑ

Αυτό που προκύπτει σαν συμπέρασμα από την εργασία είναι ότι η προσεγγιστική μέθοδος του Fehlberg που χρησιμοποιήσαμε βγάζει καλύτερα αποτελέσματα και αυτό εξηγείται από το λόγο ότι το απόλυτο σφάλμα μας είναι μικρότερο σε σχέση με τα απόλυτα σφάλματα των άλλων μεθόδων. Είναι σίγουρο πως αν μεγαλώσει η τάξη ή ελαττώσουμε το βήμα μας και αυξήσουμε τον αριθμό των υποδιαστημάτων τότε θα έχουμε καλύτερα αποτελέσματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Πανεπιστημιακές Παραδόσεις ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΙΙ της επίκουρης καθηγήτριας Μ. Γουσίδου-Κουτίτα(1993)
2. Αριθμητική Ανάλυση Μαρία Χ. Γουσίδου- Κουτίτα Εκδόσεις Χριστοδουλίδη(2004)
3. Αριθμητικές Μέθοδοι Με Εφαρμογές Κανονικών(Συνήθων) Και Μερικών Διαφορικών Εξισώσεων Μαρία Γουσίδου-Κουτίτα(2009)
4. Αριθμητικές Μέθοδοι Με Εφαρμογές Στη Θεωρία Ελέγχου Μαρία Γουσίδου-Κουτίτα(2006)
5. Διαφορικές Εξισώσεις Θωμά Κυβεντίδη (2004)
6. Συνήθεις Διαφορικές Εξισώσεις Στέφανος Τραγανάς Πανεπιστημιακές Εκδόσεις Κρήτης(2005)
7. Εισαγωγή στις Διαφορικές Εξισώσεις Χ. Φίλος(1990)
8. L.Lapidus and Luus R, 'Optimal Control of Engineering Process.' Random House (Blaisdell), New York, 1967

ΙΝΤΕΡΝΕΤ

1. Σημειώσεις του κ. Νικόλαου Καραμπετάκη από το μεταπτυχιακό μάθημα Εισαγωγή στο Matlab που βρίσκονται στο eclass.auth.gr

ΠΑΡΑΡΤΗΜΑ

Στο σημείο αυτό παρουσιάζονται όλα τα προγράμματα που βασίζονται στους αλγόριθμους που περιέχονται στο κύριο μέρος της εργασίας υπολογίζουν τις προσεγγίσεις όλων των μεθόδων στο γνωστό πρόβλημα των αρχικών τιμών. Όλα αυτά τα προγράμματα κατασκευάστηκαν εξ' ολοκλήρου στο matlabR2007a από τον συγγραφέα της παρούσας εργασίας. Ακολουθούν την ροή της εργασίας και δεν εμφανίζονται σχόλια.

1.EULER(1,1)(ΠΑΡΑΔΕΙΓΜΑ 2.1.1 h=0.25)

```
format('long')
f=inline('2*x-y','x','y');
dsolve('Dy=2*x-y','y(0)=1','x')
g=inline('-2+2*x+3*exp(-x)','x')
h=0.25;
y(1)=1;
i=0:0.25:1;
nstep=5;
for n=1:nstep
   y(n+1)=y(n)+h*f(i(n),y(n));
end
for i=0:0.25:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.25:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g((j-1)/4)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
ans =
-2+2*x+3*exp(-x)
g =
     Inline function:
     g(x) = -2+2*x+3*exp(-x)
x0=
     0
x0.25=
  0.250000000000000
x0.5=
  0.500000000000000
x0.75=
  0.750000000000000
x1=
     1
```

```
y1=
     1
y2=
   0.750000000000000
y3=
   0.687500000000000
y4=
   0.765625000000000
y5=
   0.949218750000000
g0=
     1
g0.25=
   0.836402349214215
g0.5=
   0.819591979137900
g0.75=
   0.917099658223044
g1=
   1.103638323514327
e1=
     0
e2=
   0.086402349214215
e3=
   0.132091979137900
e4=
   0.151474658223044
e5=
   0.154419573514327
```

2.EULER(1,1)(ΠΑΡΑΔΕΙΓΜΑ 2.1.1 h=0.1)

```
format('long')
f=inline('2*x-y','x','y');
g=inline('-2+2*x+3*exp(-x)','x')
h=0.1;
y(1)=1;
i=0:0.1:1;
nstep=11;
for n=1:nstep
    y(n+1)=y(n)+h*f(i(n),y(n));
end
for i=0:0.10:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
    disp (['y' num2str(n) '='])
    disp(y(n))
end
for i=0:0.1:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
    e=abs(g((j-1)/10)-y(j));
    disp (['e' num2str(j) '='])
    disp(e)
```

```
end
```

Η εκτέλεση του προγράμματος δίνει τα εξής αποτελέσματα:

```
g =
    Inline function:
    g(x) = -2+2*x+3*exp(-x)
x0=
    0
x0.1=
  0.100000000000000
x0.2=
  0.200000000000000
x0.3=
  0.300000000000000
x0.4=
  0.400000000000000
x0.5=
  0.500000000000000
x0.6=
  0.600000000000000
x0.7=
  0.700000000000000
x0.8=
  0.800000000000000
x0.9=
  0.900000000000000
x1=
    1
y1=
    1
y2=
  0.900000000000000
y3=
  0.830000000000000
y4=
  0.787000000000000
y5=
  0.768300000000000
y6=
  0.771470000000000
y7=
  0.794323000000000
y8=
  0.834890700000000
y9=
  0.891401630000000
y10=
  0.962261467000000
y11=
  1.046035320300000
g0=
    1
g0.1=
  0.914512254107879
g0.2=
  0.856192259233945
g0.3=
  0.822454662045153
g0.4=
```

```
  0.810960138106918
g0.5=
  0.819591979137900
g0.6=
  0.846434908282079
g0.7=
  0.889755911374229
g0.8=
  0.947986892351665
g0.9=
  1.019708979221797
g1=
  1.103638323514327
e1=
    0
e2=
  0.014512254107879
e3=
  0.026192259233945
e4=
  0.035454662045154
e5=
  0.042660138106918
e6=
  0.048121979137900
e7=
  0.052111908282079
e8=
  0.054865211374229
e9=
  0.056585262351665
e10=
  0.057447512221797
e11=
  0.057603003214327
```

3.EULER(1,1)(ΠΑΡΑΔΕΙΓΜΑ 2.1.2 h=0.25)

```
format('long')
f=inline('x*y+1','x','y');
g=inline('(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)','x')
h=0.25;
y(1)=1;
i=0:0.25:1;
nstep=5;
for n=1:nstep
   y(n+1)=y(n)+h*f(i(n),y(n));
end
for i=0:0.25:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.25:1
disp (['g' num2str(i) '=' ])
disp(g(i))
```

```
end
for j=1:1:5
   e=abs(g((j-1)/4)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
g =
     Inline function:
     g(x) = (1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)
x0=
     0
x0.25=
   0.250000000000000
x0.5=
   0.500000000000000
x0.75=
   0.750000000000000
x1=
     1
y1=
     1
y2=
   1.250000000000000
y3=
   1.578125000000000
y4=
   2.025390625000000
y5=
   2.655151367187500
g0=
     1
g0.25=
   1.287017430346069
g0.5=
   1.676974972518977
g0.75=
   2.232585046680449
g1=
   3.059407405342577
e1=
     0
e2=
   0.037017430346069
e3=
   0.098849972518977
e4=
   0.207194421680449
e5=
   0.404256038155077
```

4.EULER(1,1)(ΠΑΡΑΔΕΙΓΜΑ 2.1.2 h=0.1)

```
format('long')
f=inline('x*y+1','x','y');
dsolve('Dy=x*y+1','y(0)=1','x')
g=inline('(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)','x')
h=0.1;
```

```
y(1)=1;
i=0:0.1:1;
nstep=11;
for n=1:nstep
   y(n+1)=y(n)+h*f(i(n),y(n));
end
for i=0:0.1:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.1:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g((j-1)/10)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
ans =
(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
g =
    Inline function:
    g(x) = (1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)
x0=
    0
x0.1=
  0.100000000000000
x0.2=
  0.200000000000000
x0.3=
  0.300000000000000
x0.4=
  0.400000000000000
x0.5=
  0.500000000000000
x0.6=
  0.600000000000000
x0.7=
  0.700000000000000
x0.8=
  0.800000000000000
x0.9=
  0.900000000000000
x1=
    1
y1=
    1
y2=
  1.100000000000000
y3=
  1.211000000000000
y4=
```

```
  1.335220000000000
y5=
  1.475276600000000
y6=
  1.634287664000000
y7=
  1.816002047200000
y8=
  2.024962170032000
y9=
  2.266709521934240
y10=
  2.548046283688980
y11=
  2.877370449220988
g0=
  1
g0.1=
  1.105346521812841
g0.2=
  1.222889462475293
g0.3=
  1.355191963766034
g0.4=
  1.505318952970661
g0.5=
  1.676974972518977
g0.6=
  1.874678991977672
g0.7=
  2.103988318400775
g0.8=
  2.371787769675660
g0.9=
  2.686665853619038
g1=
  3.059407405342577
e1=
    0
e2=
  0.005346521812841
e3=
  0.011889462475293
e4=
  0.019971963766034
e5=
  0.030042352970660
e6=
  0.042687308518977
e7=
  0.058676944777672
e8=
  0.079026148368775
e9=
  0.105078247741420
e10=
  0.138619569930059
e11=
  0.182036956121589
```

5.RUNGE-KUTTA(2,2)(ΠΑΡΑΔΕΙΓΜΑ 2.1.3 h=0.25)

```
format('long')
f=inline('2*x-y','x','y');
dsolve('Dy=2*x-y','y(0)=1','x')
g=inline('-2+2*x+3*exp(-x)','x')
w1=0;
w2=1;
c1=0;
c2=1/2;
a21=1/2;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
y(n+1)=y(n)+w1*k1+w2*k2;
end
for i=0:0.25:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.25:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g((j-1)/4)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
ans =
 -2+2*x+3*exp(-x)
g =
     Inline function:
     g(x) = -2+2*x+3*exp(-x)
x0=
     0
x0.25=
   0.250000000000000
x0.5=
   0.500000000000000
x0.75=
   0.750000000000000
x1=
     1
y1=
     1
y2=
   0.843750000000000
y3=
```

```
  0.831054687500000
y4=
  0.930511474609375
y5=
  1.117587089538574
g0=
    1
g0.25=
  0.836402349214215
g0.5=
  0.819591979137900
g0.75=
  0.917099658223044
g1=
  1.103638323514327
e1=
    0
e2=
  0.007347650785785
e3=
  0.011462708362100
e4=
  0.013411816386331
e5=
  0.013948766024247
```

6.RUNGE-KUTTA(2,2)(ΠΑΡΑΔΕΙΓΜΑ 2.1.3 h=0.1)

```
format('long')
f=inline('2*x-y','x','y');
dsolve('Dy=2*x-y','y(0)=1','x')
g=inline('-2+2*x+3*exp(-x)','x')
w1=0;
w2=1;
c1=0;
c2=1/2;
a21=1/2;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
y(n+1)=y(n)+w1*k1+w2*k2;
end
for i=0:0.1:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.1:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g((j-1)/10)-y(j));
```

```
    disp (['e' num2str(j) '='])
    disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
ans =
-2+2*x+3*exp(-x)
g =
     Inline function:
     g(x) = -2+2*x+3*exp(-x)
x0=
     0
x0.1=
   0.100000000000000
x0.2=
   0.200000000000000
x0.3=
   0.300000000000000
x0.4=
   0.400000000000000
x0.5=
   0.500000000000000
x0.6=
   0.600000000000000
x0.7=
   0.700000000000000
x0.8=
   0.800000000000000
x0.9=
   0.900000000000000
x1=
   1
y1=
   1
y2=
   0.915000000000000
y3=
   0.857075000000000
y4=
   0.823652875000000
y5=
   0.812405851875000
y6=
   0.821227295946875
y7=
   0.848210702831922
y8=
   0.891630686062889
y9=
   0.949925770886915
y10=
   1.021682822652658
y11=
   1.105622954500656
g0=
     1
g0.1=
   0.914512254107879
g0.2=
```

```
  0.856192259233945
g0.3=
  0.822454662045153
g0.4=
  0.810960138106918
g0.5=
  0.819591979137900
g0.6=
  0.846434908282079
g0.7=
  0.889755911374229
g0.8=
  0.947986892351665
g0.9=
  1.019708979221797
g1=
  1.103638323514327
e1=
    0
e2=
   4.877458921213052e-004
e3=
   8.827407660545461e-004
e4=
  0.001198212954846
e5=
  0.001445713768082
e6=
  0.001635316808975
e7=
  0.001775794549842
e8=
  0.001874774688661
e9=
  0.001938878535250
e10=
  0.001973843430861
e11=
  0.001984630986329
```

7.RUNGE-KUTTA(2,2)(ΠΑΡΑΔΕΙΓΜΑ 2.1.4 h=0.25)

```
format('long')
f=inline('x*y+1','x','y');
dsolve('Dy=x*y+1','y(0)=1','x')
g=inline('(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)','x')
w1=0;
w2=1;
c1=0;
c2=1/2;
a21=1/2;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
y(n+1)=y(n)+w1*k1+w2*k2;
```

```
end
for i=0:0.25:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.25:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g((j-1)/4)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
ans =
(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)
g =
    Inline function:

    g(x) = (1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)
x0=
0
x0.25=
  0.250000000000000
x0.5=
  0.500000000000000
x0.75=
  0.750000000000000
x1=
    1
y1=
    1
y2=
  1.285156250000000
y3=
  1.671123504638672
y4=
  2.218087367713451
y5=
  3.026125849246455
g0=
    1
g0.25=
  1.287017430346069
g0.5=
  1.676974972518977
g0.75=
  2.232585046680449
```

```
g1=
   3.059407405342577
e1=
     0
e2=
   0.001861180346069
e3=
   0.005851467880305
e4=
   0.014497678966998
e5=
   0.033281556096122
```

8.RUNGE-KUTTA(2,2)(ΠΑΡΑΔΕΙΓΜΑ 2.1.4 h=0.1)

```
format('long')
f=inline('x*y+1','x','y');
dsolve('Dy=x*y+1','y(0)=1','x')
g=inline('(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)','x')
w1=0;
w2=1;
c1=0;
c2=1/2;
a21=1/2;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
y(n+1)=y(n)+w1*k1+w2*k2;
end
for i=0:0.1:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.1:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g((j-1)/10)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
ans =(1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)

 g =

   Inline function:
```

g(x) = (1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+1)*exp(1/2*x^2)

x0=
0
x0.1=
0.100000000000000
x0.2=
0.200000000000000
x0.3=
0.300000000000000
x0.4=
0.400000000000000
x0.5=
0.500000000000000
x0.6=
0.600000000000000
x0.7=
0.700000000000000
x0.8=
0.800000000000000
x0.9=
0.900000000000000
x1=
1
y1=
1
y2=
1.105250000000000
y3=
1.222661643750000
y4=
1.354783850254687
y5=
1.504662546534985
y6=
1.675976557420941
y7=
1.873209735845547
y8=
2.101871127660406
y9=
2.368778873945045
y10=
2.682428926401787
y11=
3.053477058070325
g0=
1
g0.1=
1.105346521812841
g0.2=

```
  1.222889462475293
g0.3=
  1.355191963766034
g0.4=
  1.505318952970661
g0.5=
  1.676974972518977
g0.6=
  1.874678991977672
g0.7=
  2.103988318400775
g0.8=
  2.371787769675660
g0.9=
  2.686665853619038
g1=
  3.059407405342577
e1=
    0
e2=
  9.652181284081074e-005
e3=
   2.278187252928721e-004
e4=
   4.081135113462864e-004
e5=
   6.564064356753274e-004
e6=
   9.984150980357764e-004
e7=
  0.001469256132126
e8=
  0.002117190740369
e9=
  0.003008895730615
e10=
  0.004236927217251
e11=
  0.005930347272252
```

9.RUNGE-KUTTA(3,3)(ΠΑΡΑΔΕΙΓΜΑ 3.1.1 h=0.25)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=3/8;
w2=2/3;
w3=1/6;
c1=0;
c2=1/2;
c3=1;
a21=1/2;
```

```
a31=-1;
a32=2;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3;
end
for i=1:0.25:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.25:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =

    Inline function:

    f(x,y) = -x*y^2

g =

    Inline function:

    g(x) = 2/(2+x^2)

x1=
     1
x1.25=
   1.250000000000000
x1.5=
   1.500000000000000
x1.75=
   1.750000000000000
x2=
     2
y1=
     1
y2=
   0.970011393229167
```

```
y3=
  0.877148621911529
y4=
  0.751505082621579
y5=
  0.623355468444529
g1=
  0.666666666666667
g1.25=
  0.561403508771930
g1.5=
  0.470588235294118
g1.75=
  0.395061728395062
g2=
  0.333333333333333
e1=
  0.333333333333333
e2=
  0.636678059895833
e3=
  0.695330440093347
e4=
  0.640393971510468
e5=
  0.549281394370455
```

10.RUNGE-KUTTA(3,3)(ΠΑΡΑΔΕΙΓΜΑ 3.1.1 h=0.1)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=3/8;
w2=2/3;
w3=1/6;
c1=0;
c2=1/2;
c3=1;
a21=1/2;
a31=-1;
a32=2;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3;
end
for i=1:0.1:2
disp (['x' num2str(i) '='])
disp(i)
```

```
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.1:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:
    f(x,y) = -x*y^2
g =
    Inline function:
    g(x) = 2/(2+x^2)
x1=
    1
x1.1=
  1.100000000000000
x1.2=
  1.200000000000000
x1.3=
  1.300000000000000
x1.4=
  1.400000000000000
x1.5=
  1.500000000000000
x1.6=
  1.600000000000000
x1.7=
  1.700000000000000
x1.8=
  1.800000000000000
x1.9=
  1.900000000000000
x2=
    2
y1=
    1
y2=
  0.995033166666667
y3=
  0.978345516662117
y4=
  0.951006922658594
y5=
  0.914725718014457
y6=
  0.871593973309766
y7=
  0.823813318698911
y8=
```

```
  0.773460831682224
y9=
  0.722329481693141
y10=
  0.671849519097985
y11=
  0.623077078895250
g1=
  0.666666666666667
g1.1=
  0.623052959501558
g1.2=
  0.581395348837209
g1.3=
  0.542005420054200
g1.4=
  0.505050505050505
g1.5=
  0.470588235294118
g1.6=
  0.438596491228070
g1.7=
  0.408997955010225
g1.8=
  0.381679389312977
g1.9=
  0.356506238859180
g2=
  0.333333333333333
e1=
  0.333333333333333
e2=
  0.661699833333333
e3=
  0.796527334843935
e4=
  0.839895811547483
e5=
  0.840651643940383
e6=
  0.818962394362398
e7=
  0.784597632424401
e8=
  0.743157801379194
e9=
  0.698233096150972
e10=
  0.652241675960730
e11=
  0.606816916293624
```

11.NYSTROM(3,3)(ΠΑΡΑΔΕΙΓΜΑ 3.1.2 h=0.25)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/4;
w2=3/8;
```

```
w3=3/8;
c1=0;
c2=2/3;
c3=2/3;
a21=2/3;
a31=0;
a32=2/3;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3;
end
for i=1:0.25:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.25:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2
g =
    Inline function:

    g(x) = 2/(2+x^2)
x1=
    1
x1.25=
  1.250000000000000
x1.5=
  1.500000000000000
x1.75=
  1.750000000000000
x2=
    2
y1=
```

```
    1
y2=
   0.969605999228395
y3=
   0.888788475869516
y4=
   0.780408616697521
y5=
   0.66656602025201
g1=
   0.666666666666667
g1.25=
   0.561403508771930
g1.5=
   0.470588235294118
g1.75=
   0.395061728395062
g2=
   0.333333333333333
e1=
   0.333333333333333
e2=
   0.636272665895062
e3=
   0.706970294051334
e4=
   0.669297505586410
e5=
   0.592491946177941
```

12.NYSTROM(3,3)(ΠΑΡΑΔΕΙΓΜΑ 3.1.2 h=0.1)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/4;
w2=3/8;
w3=3/8;
c1=0;
c2=2/3;
c3=2/3;
a21=2/3;
a31=0;
a32=2/3;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3;
```

```
end
for i=1:0.1:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.1:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
     Inline function:
     f(x,y) = -x*y^2

g =
    Inline function:

     g(x) = 2/(2+x^2)

x1=
     1
x1.1=
   1.100000000000000
x1.2=
   1.200000000000000
x1.3=
   1.300000000000000
x1.4=
   1.400000000000000
x1.5=
   1.500000000000000
x1.6=
   1.600000000000000
x1.7=
   1.700000000000000
x1.8=
   1.800000000000000
x1.9=
   1.900000000000000
x2=
     2
y1=
     1
y2=
   0.995022172839506
```

y3=
  0.980387238129322
y4=
  0.956931470376676
y5=
  0.925919033349061
y6=
  0.888882060498600
y7=
  0.847451135371869
y8=
  0.803206620187101
y9=
  0.757569472732688
y10=
  0.711737028168946
y11=
  0.666659092863906
g1=
  0.666666666666667
g1.1=
  0.623052959501558
g1.2=
  0.581395348837209
g1.3=
  0.542005420054200
g1.4=
  0.505050505050505
g1.5=
  0.470588235294118
g1.6=
  0.438596491228070
g1.7=
  0.408997955010225
g1.8=
  0.381679389312977
g1.9=
  0.356506238859180
g2=
  0.333333333333333
e1=
  0.333333333333333
e2=
  0.661688839506173
e3=
  0.798569056311140
e4=
  0.845820359265565
e5=
  0.851844959274987
e6=

```
  0.836250481551231
e7=
  0.808235449097360
e8=
  0.772903589884071
e9=
  0.733473087190520
e10=
  0.692129185031691
e11=
  0.650398930262280
```

13.HEWN(3,3)(ΠΑΡΑΔΕΙΓΜΑ 3.1.3 h=0.25)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/4;
w2=0;
w3=3/4;
c1=0;
c2=1/3;
c3=1/3;
a21=1/3;
a31=0;
a32=2/3;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3;
end
for i=1:0.25:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.25:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
```

```
    Inline function:
    f(x,y) = -x*y^2
g =
    Inline function:
    g(x) = 2/(2+x^2)
x1=
    1
x1.25=
  1.250000000000000
x1.5=
  1.500000000000000
x1.75=
  1.750000000000000
x2=
    2
y1=
    1
y2=
  0.984806013695988
y3=
  0.915233351982195
y4=
  0.811902342605009
y5=
  0.697873163942227
g1=
  0.666666666666667
g1.25=
  0.561403508771930
g1.5=
  0.470588235294118
g1.75=
  0.395061728395062
g2=
  0.333333333333333
e1=
  0.333333333333333
e2=
  0.651472680362654
e3=
  0.733415170164013
e4=
  0.700791231493898
e5=
  0.623799089868153
```

14.ΗΕΨΝ(3,3)(ΠΑΡΑΔΕΙΓΜΑ 3.1.3 h=0.1)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/4;
w2=0;
w3=3/4;
c1=0;
c2=1/3;
c3=1/3;
a21=1/3;
```

```
a31=0;
a32=2/3;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3;
end
for i=1:0.1:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.1:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = 2/(2+x^2)

x1=
    1
x1.1=
  1.100000000000000
x1.2=
  1.200000000000000
x1.3=
  1.300000000000000
x1.4=
  1.400000000000000
x1.5=
  1.500000000000000
x1.6=
  1.600000000000000
x1.7=
```

```
  1.700000000000000
x1.8=
  1.800000000000000
x1.9=
  1.900000000000000
x2=
    2
y1=
    1
y2=
  0.997511098765432
y3=
  0.985247754115330
y4=
  0.963916707176488
y5=
  0.934685582637371
y6=
  0.899031683205091
y7=
  0.858572478318557
y8=
  0.814910049211077
y9=
  0.769511316784970
y10=
  0.723632680750224
y11=
  0.678286626453273
g1=
  0.666666666666667
g1.1=
  0.623052959501558
g1.2=
  0.581395348837209
g1.3=
  0.542005420054200
g1.4=
  0.505050505050505
g1.5=
  0.470588235294118
g1.6=
  0.438596491228070
g1.7=
  0.408997955010225
g1.8=
  0.381679389312977
g1.9=
  0.356506238859180
g2=
  0.333333333333333
```

```
e1=
  0.333333333333333
e2=
  0.664177765432099
e3=
  0.803429572297148
e4=
  0.852805596065377
e5=
  0.860611508563297
e6=
  0.846400104257723
e7=
  0.819356792044048
e8=
  0.784607018908047
e9=
  0.745414931242802
e10=
  0.704024837612970
e11=
  0.662026463851647
```

15.RUNGE KUTTA(4,4)(ΠΑΡΑΔΕΙΓΜΑ 3.2.1 h=0.25)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/6;
w2=1/3;
w3=1/3;
w4=1/6;
c1=0;
c2=1/2;
c3=1/2;
c4=1;
a21=1/2;
a31=0;
a32=1/2;
a41=0;
a42=0;
a43=1;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c4*h,y(n)+a41*k1+a42*k2+a43*k3);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4;
end
for i=1:0.25:2
```

```
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.25:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = 2/(2+x^2)

x1=
    1
x1.25=
  1.250000000000000
x1.5=
  1.500000000000000
x1.75=
  1.750000000000000
x2=
    2
y1=
    1
y2=
  0.969694281772415
y3=
  0.888880444891296
y4=
  0.780477845927160
y5=
  0.666661297755045
g1=
  0.666666666666667
g1.25=
  0.561403508771930
g1.5=
  0.470588235294118
g1.75=
```

```
  0.395061728395062
g2=
  0.333333333333333
e1=
  0.333333333333333
e2=
  0.636360948439081
e3=
  0.707062263073114
e4=
  0.669366734816049
e5=
  0.592587223680971
```

16. RUNGE KUTTA(4,4)(ΠΑΡΑΔΕΙΓΜΑ 3.2.1 h=0.1)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/6;
w2=1/3;
w3=1/3;
w4=1/6;
c1=0;
c2=1/2;
c3=1/2;
c4=1;
a21=1/2;
a31=0;
a32=1/2;
a41=0;
a42=0;
a43=1;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c4*h,y(n)+a41*k1+a42*k2+a43*k3);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4;
end
for i=1:0.1:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.1:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
```

```
    e=abs(g(j)-y(j));
    disp (['e' num2str(j) '='])
    disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
     Inline function:

     f(x,y) = -x*y^2

g =

     Inline function:

     g(x) = 2/(2+x^2)

x1=
     1
x1.1=
   1.100000000000000
x1.2=
   1.200000000000000
x1.3=
   1.300000000000000
x1.4=
   1.400000000000000
x1.5=
   1.500000000000000
x1.6=
   1.600000000000000
x1.7=
   1.700000000000000
x1.8=
   1.800000000000000
x1.9=
   1.900000000000000
x2=
     2
y1=
     1
y2=
   0.995024865102607
y3=
   0.980392116100969
y4=
   0.956937715060499
y5=
   0.925925796601353
y6=
   0.888888723683622
y7=
   0.847457444393658
y8=
```

0.803212673650688
y9=
0.757575606404068
y10=
0.711743664631487
y11=
0.666666613089039
g1=
0.666666666666667
g1.1=
0.623052959501558
g1.2=
0.581395348837209
g1.3=
0.542005420054200
g1.4=
0.505050505050505
g1.5=
0.470588235294118
g1.6=
0.438596491228070
g1.7=
0.408997955010225
g1.8=
0.381679389312977
g1.9=
0.356506238859180
g2=
0.333333333333333
e1=
0.333333333333333
e2=
0.661691531769274
e3=
0.798573934282788
e4=
0.845826603949388
e5=
0.851851722527279
e6=
0.836257144736254
e7=
0.808241758119148
e8=
0.772909643347658
e9=
0.733479220861900
e10=
0.692135821494232
e11=
0.650406450487413

17.ΚUTTA ΤΥΠΟΣ(4,4)(ΠΑΡΑΔΕΙΓΜΑ 3.2.2 h=0.25)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/8;
w2=3/8;
w3=3/8;
w4=1/8;
c1=0;
c2=1/3;
c3=2/3;
c4=1;
a21=1/3;
a31=-1/3;
a32=1;
a41=1;
a42=-1;
a43=1;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c4*h,y(n)+a41*k1+a42*k2+a43*k3);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4;
end
for i=1:0.25:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.25:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = 2/(2+x^2)
```

```
x1=
     1
x1.25=
   1.250000000000000
x1.5=
   1.500000000000000
x1.75=
   1.750000000000000
x2=
     2
y1=
     1
y2=
   0.969690082001976
y3=
   0.888866928819342
y4=
   0.780457844383869
y5=
   0.666640176087096
g1=
   0.666666666666667
g1.25=
   0.561403508771930
g1.5=
   0.470588235294118
g1.75=
   0.395061728395062
g2=
   0.333333333333333
e1=
   0.333333333333333
e2=
   0.636356748668643
e3=
   0.707048747001160
e4=
   0.669346733272758
e5=
   0.592566102013022
```

18.ΚUTTA ΤΥΠΟΣ(4,4)(ΠΑΡΑΔΕΙΓΜΑ 3.2.2 h=0.1)

```
format('long')
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('2/(2+x^2)','x')
w1=1/8;
w2=3/8;
w3=3/8;
w4=1/8;
c1=0;
c2=1/3;
```

```
c3=2/3;
c4=1;
a21=1/3;
a31=-1/3;
a32=1;
a41=1;
a42=-1;
a43=1;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c4*h,y(n)+a41*k1+a42*k2+a43*k3);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4;
end
for i=1:0.1:2
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=1:0.1:2
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g(j)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = 2/(2+x^2)

x1=
    1
x1.1=
  1.100000000000000
x1.2=
  1.200000000000000
x1.3=
  1.300000000000000
x1.4=
  1.400000000000000
```

x1.5=
  1.500000000000000
x1.6=
  1.600000000000000
x1.7=
  1.700000000000000
x1.8=
  1.800000000000000
x1.9=
  1.900000000000000
x2=
    2
y1=
    1
y2=
  0.995024847774700
y3=
  0.980392049223310
y4=
  0.956937575228808
y5=
  0.925925573217522
y6=
  0.888888419154680
y7=
  0.847457071166520
y8=
  0.803212249762128
y9=
  0.757575151244819
y10=
  0.711743195936359
y11=
  0.666666145369993
g1=
  0.666666666666667
g1.1=
  0.623052959501558
g1.2=
  0.581395348837209
g1.3=
  0.542005420054200
g1.4=
  0.505050505050505
g1.5=
  0.470588235294118
g1.6=
  0.438596491228070
g1.7=
  0.408997955010225
g1.8=

```
  0.381679389312977
g1.9=
  0.356506238859180
g2=
  0.333333333333333
e1=
  0.333333333333333
e2=
  0.661691514441366
e3=
  0.798573867405128
e4=
  0.845826464117697
e5=
  0.851851499143448
e6=
  0.836256840207311
e7=
  0.808241384892010
e8=
  0.772909219459097
e9=
  0.733478765702650
e10=
  0.692135352799104
e11=
  0.650405982768367
```

19.NYSTROM(5,6)(ΠΑΡΑΔΕΙΓΜΑ 4.1.1 h=0.25)

```
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('-1+x+2*exp(-x)','x')
format('long')
c1=0;
c2=1/3;
c3=2/5;
c4=1;
c5=2/3;
c6=4/5;
w1=23/192;
w2=0;
w3=125/192;
w4=-81/192;
w5=125/192;
a21=1/3;
a31=4/25;
a32=6/25;
a41=1/4;
a42=-12/4;
a43=15/4;
a51=6/81;
a52=90/81;
a53=-50/81;
a54=8/81;
```

```
a61=6/75;
a62=36/75;
a63=10/75;
a64=8/75;
a65=0;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c5*h,y(n)+a41*k1+a42*k2+a43*k3);
k5=h*f(i(n)+c5*h,y(n)+a51*k1+a52*k2+a53*k3+a54*k4);
k6=h*f(i(n)+c6*h,y(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4+w5*k5;
end
for i=0:0.25:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.25:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:6
   e=abs(g((j-1)/100)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = -1+x+2*exp(-x)

x0=
    0
x0.25=
  0.250000000000000
x0.5=
  0.500000000000000
x0.75=
  0.750000000000000
x1=
    1
y1=
```

```
    1
y2=
  0.973921268121264
y3=
  0.892778599719703
y4=
  0.779891810573625
y5=
  0.660448711605436
g0=
    1
g0.25=
  0.807601566142810
g0.5=
  0.713061319425267
g0.75=
  0.694733105482029
g1=
  0.735758882342885
e1=
    0
e2=
  0.016178399377072
e3=
  0.087618746893808
e4=
  0.190999256523391
e5=
  0.301130166699210
```

20.NYSTROM(5,6)(ΠΑΡΑΔΕΙΓΜΑ 4.1.1 h=0.1)

```
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('-1+x+2*exp(-x)','x')
format('long')
c1=0;
c2=1/3;
c3=2/5;
c4=1;
c5=2/3;
c6=4/5;
w1=23/192;
w2=0;
w3=125/192;
w4=-81/192;
w5=125/192;
a21=1/3;
a31=4/25;
a32=6/25;
a41=1/4;
a42=-12/4;
a43=15/4;
a51=6/81;
```

```
a52=90/81;
a53=-50/81;
a54=8/81;
a61=6/75;
a62=36/75;
a63=10/75;
a64=8/75;
a65=0;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c5*h,y(n)+a41*k1+a42*k2+a43*k3);
k5=h*f(i(n)+c5*h,y(n)+a51*k1+a52*k2+a53*k3+a54*k4);
k6=h*f(i(n)+c6*h,y(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4+w5*k5;
end
for i=0:0.1:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.1:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:10
   e=abs(g((j-1)/100)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = -1+x+2*exp(-x)

x0=
    0
x0.1=
  0.100000000000000
x0.2=
  0.200000000000000
x0.3=
  0.300000000000000
```

x0.4=
  0.400000000000000
x0.5=
  0.500000000000000
x0.6=
  0.600000000000000
x0.7=
  0.700000000000000
x0.8=
  0.800000000000000
x0.9=
  0.900000000000000
x1=
    1
y1=
    1
y2=
  0.995860694497265
y3=
  0.981909066057179
y4=
  0.958876358832192
y5=
  0.927970733149774
y6=
  0.890722321728846
y7=
  0.848805325158573
y8=
  0.803872723532910
y9=
  0.757428697817751
y10=
  0.710749250282059
y11=
  0.664848732149450
g0=
    1
g0.1=
  0.909674836071919
g0.2=
  0.837461506155964
g0.3=
  0.781636441363436
g0.4=
  0.740640092071279
g0.5=
  0.713061319425267
g0.6=
  0.697623272188053
g0.7=

```
  0.693170607582819
g0.8=
  0.698657928234443
g0.9=
  0.713139319481198
g1=
  0.735758882342885
e1=
    0
e2=
  0.005761026998929
e3=
  0.001511719443668
e4=
  0.012014708264824
e5=
  0.033608145154872
e6=
  0.061736527272582
e7=
  0.094723742009925
e8=
  0.130914916278987
e9=
  0.168803994955520
e10=
  0.207113120260398
```

21.FEHLBERG(5,6)(ΠΑΡΑΔΕΙΓΜΑ 4.1.2 h=0.25)

```
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('-1+x+2*exp(-x)','x')
format('long')
c1=0;
c2=1/6;
c3=4/15;
c4=2/3;
c5=4/5;
c6=1;
w1=31/384;
w2=0;
w3=1125/2816;
w4=9/32;
w5=125/768;
w6=5/66;
a21=1/6;
a31=4/75;
a32=16/75;
a41=5/6;
a42=-8/3;
a43=5/2;
a51=-8/5;
a52=144/25;
a53=-4;
```

```
a54=16/25;
a61=361/320;
a62=-18/5;
a63=407/128;
a64=-11/80;
a65=55/128;
h=0.25;
i=0:0.25:1;
nstep=5;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c5*h,y(n)+a41*k1+a42*k2+a43*k3);
k5=h*f(i(n)+c5*h,y(n)+a51*k1+a52*k2+a53*k3+a54*k4);
k6=h*f(i(n)+c6*h,y(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4+w5*k5+w6*k6;
end
for i=0:0.25:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.25:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:5
   e=abs(g((j-1)/100)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = -1+x+2*exp(-x)

x0=
    0
x0.25=
  0.250000000000000
x0.5=
  0.500000000000000
x0.75=
  0.750000000000000
x1=
```

```
    1
y1=
    1
y2=
  0.967487258166394
y3=
  0.885180544753583
y4=
  0.776203891812263
y5=
  0.662502743089561
g0=
    1
g0.25=
  0.807601566142810
g0.5=
  0.713061319425267
g0.75=
  0.694733105482029
g1=
  0.735758882342885
e1=
    0
e2=
  0.022612409331943
e3=
  0.095216801859928
e4=
  0.194687175284753
e5=
  0.299076135215086
```

22.FEHLBERG(5,6)(ΠΑΡΑΔΕΙΓΜΑ 4.1.2 h=0.1)

```
f=inline('-x*y^2','x','y')
dsolve('Dy=-x*y^2','y(0)=1','x');
g=inline('-1+x+2*exp(-x)','x')
format('long')
c1=0;
c2=1/6;
c3=4/15;
c4=2/3;
c5=4/5;
c6=1;
w1=31/384;
w2=0;
w3=1125/2816;
w4=9/32;
w5=125/768;
w6=5/66;
a21=1/6;
a31=4/75;
a32=16/75;
a41=5/6;
```

```
a42=-8/3;
a43=5/2;
a51=-8/5;
a52=144/25;
a53=-4;
a54=16/25;
a61=361/320;
a62=-18/5;
a63=407/128;
a64=-11/80;
a65=55/128;
h=0.1;
i=0:0.1:1;
nstep=11;
y(1)=1;
for n=1:nstep
k1=h*f(i(n)+c1*h,y(n));
k2=h*f(i(n)+c2*h,y(n)+a21*k1);
k3=h*f(i(n)+c3*h,y(n)+a31*k1+a32*k2);
k4=h*f(i(n)+c5*h,y(n)+a41*k1+a42*k2+a43*k3);
k5=h*f(i(n)+c5*h,y(n)+a51*k1+a52*k2+a53*k3+a54*k4);
k6=h*f(i(n)+c6*h,y(n)+a61*k1+a62*k2+a63*k3+a64*k4+a65*k5);
y(n+1)=y(n)+w1*k1+w2*k2+w3*k3+w4*k4+w5*k5+w6*k6;
end
for i=0:0.1:1
disp (['x' num2str(i) '='])
disp(i)
end
for n=1:nstep
   disp (['y' num2str(n) '='])
   disp(y(n))
end
for i=0:0.1:1
disp (['g' num2str(i) '=' ])
disp(g(i))
end
for j=1:1:11
   e=abs(g((j-1)/100)-y(j));
   disp (['e' num2str(j) '='])
   disp(e)
end
x=0:0.1:1;
subplot(1,2,1)
plot(y)
subplot(1,2,2)
plot(g(x))
```

Η εκτέλεση του προγράμματος δίνει τα αποτελέσματα:

```
f =
    Inline function:

    f(x,y) = -x*y^2

g =
    Inline function:

    g(x) = -1+x+2*exp(-x)
```

```
x0=
     0
x0.1=
   0.100000000000000
x0.2=
   0.200000000000000
x0.3=
   0.300000000000000
x0.4=
   0.400000000000000
x0.5=
   0.500000000000000
x0.6=
   0.600000000000000
x0.7=
   0.700000000000000
x0.8=
   0.800000000000000
x0.9=
   0.900000000000000
x1=
     1
y1=
     1
y2=
   0.994653456935136
y3=
   0.979671259036967
y4=
   0.955907912025130
y5=
   0.924640692609537
y6=
   0.887408700823141
y7=
   0.845843504129544
y8=
   0.801521548309486
y9=
   0.755856536283684
y10=
   0.710036821674568
y11=
   0.665002861028903
g0=
     1
g0.1=
   0.909674836071919
g0.2=
   0.837461506155964
g0.3=
```

```
  0.781636441363436
g0.4=
  0.740640092071279
g0.5=
  0.713061319425267
g0.6=
  0.697623272188053
g0.7=
  0.693170607582819
g0.8=
  0.698657928234443
g0.9=
  0.713139319481198
g1=
  0.735758882342885
e1=
    0
e2=
  0.004553789436800
e3=
   7.260875765433550e-004
e4=
  0.014983155071887
e5=
  0.036938185695109
e6=
  0.065050148178287
e7=
  0.097685563038953
e8=
  0.133266091502410
e9=
  0.170376156489587
e10=
  0.207825548867888
e11=
  0.244671975043016
```