ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Εργασία Εργαστηρίου

Μάθημα: Αριθμητική Ανάλυση ΙΙ & Εργαστήριο

Διδάσκων: Κωνσταντίνος Χρυσαφίνος

Φοιτήτρια: Ελένη Στυλιανού, ge21708

Email: elenistylianou03@live.com

Ομάδα Εργαστηρίου: Α, Τρίτη 12:45-14:15

Ερώτημα 1

Η συνάρτηση που υλοποιεί μια άμεση μέθοδο Runge-Kutta, που αντιστοιχεί σε ταμπλό του Butcher με πίνακα A και διανύσματα bhta και tau είναι η RKE, της οποίας ο κώδικας φαίνεται παρακάτω:

```
function [sol,t]=RKE(a,b,N,y0,f,A,tau,bhta)
  t=linspace(a,b,N+1);
  h=(b-a)/N;
  sol=zeros(N+1,1);
  sol(1)=y0;
  q=length(tau);
  for n=1:N
    tn=zeros(q,1);
    kn=zeros(q,1);
    for i=1:q
      tn(i)=t(n)+tau(i)*h;
      s1=0;
      for j=1:q
        s1=s1+h*A(i,j)*kn(j);
      end
      kn(i)=f(tn(i),sol(n)+s1);
    end
    s2=0;
    for i=1:q
      s2=s2+h*bhta(i)*kn(i);
    end
    sol(n+1)=sol(n)+s2;
  end
end
```

Στο MATLAB υλοποιήθηκε ο παρακάτω κώδικας για την εφαρμογή της παραπάνω μεθόδου για N=10, 20, 40, 80. Για τις προαναφερθέντες τιμές του N υπολογίζονται το μέγιστο απόλυτο σφάλμα, καθώς και η πειραματική τάξη ακρίβειας. Τα σφάλματα αποθηκεύονται στο διάνυσμα «errs» και η πειραματική τάξη ακρίβειας στο διάνυσμα «rates».

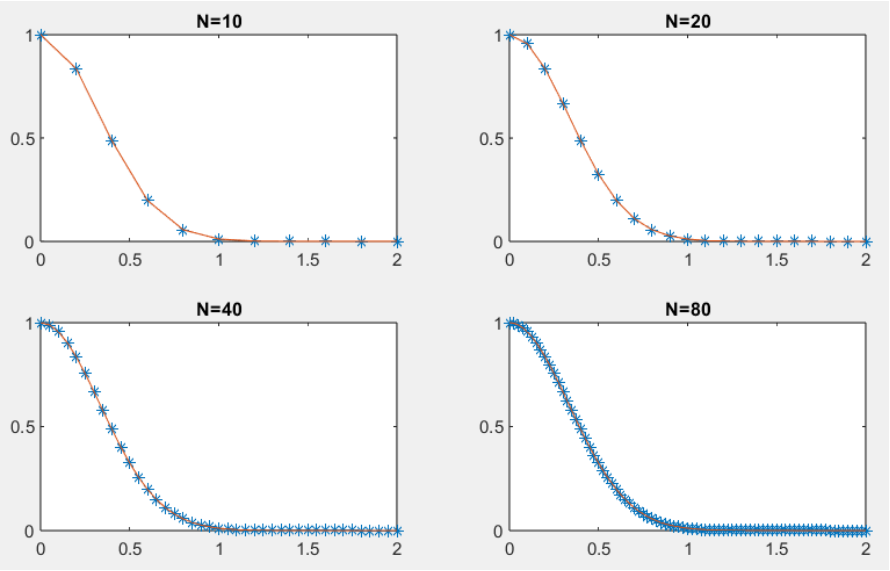
```
clear; clc;
a=0;
b=2;
y0=1;
f=@(t,y) log(y^2+1)-9*t*exp(-(9/2)*t^2)-log(exp(-9*t^2)+1);
yexact=@(t)exp(-9*t.^2 /2);
%Tableau Butcher
A=[0 0 0 0; 1/3 0 0 0; -1/3 1 0 0; 1 -1 1 0];
tau=[0; 1/3; 2/3; 1];
bhta=[1/8; 3/8; 3/8; 1/8];
Ns=[10, 20, 40, 80];
```

```
errs=zeros(length(Ns),1);
rates=zeros(length(Ns)-1,1);
for k=1:length(Ns)
   N=Ns(k);
   [Y, t]=RKE(a,b,N,y0,f,A,tau,bhta);
   yex=yexact(t);
   errs(k)=max(abs(Y(:)-yex(:)));
   subplot(2,2,k);
   plot(t,Y,'*',t,yex);
   title(["N=" num2str(N)]);
end
for i=1:length(Ns)-1
   rates(i)=log(errs(i)/errs(i+1))/log(Ns(i+1)/Ns(i));
end
errs
rates
```

Πιο κάτω παρουσιάζονται τα αποτελέσματα που εμφανίζονται στο Command Window κατά την εκτέλεση του πιο πάνω κώδικα.

```
errs =

   1.0e-03 *

    0.1457
    0.0102
    0.0006
    0.0000


rates =

    3.8429
    3.9829
    3.9945
```

Από αυτά τα αποτελέσματα παρατηρούμε πως για όλα τα N η προσέγγιση είναι πολύ κοντά στην πραγματική τιμή, αφού τα σφάλματα είναι πολύ μικρά. Επίσης, βλέπουμε πως η τάξη ακρίβειας τείνει στο 4 που είναι και η πραγματική τάξη ακρίβειας.
Τα γραφήματα που προέκυψαν:



Ερώτημα 2

Η έμμεση μέθοδος Runge-Kutta 4 σταδίων Gauss-Radau, που αντιστοιχεί στο πιο πάνω ταμπλό του Butcher, είναι η RKI (Runge-Kutta Implicit), της οποίας ο κώδικας σε MATLAB δίνεται παρακάτω:

```
function [y,t] = RKI(a,b,y0, N, A, bhta, tau ,f, Nfp)
   t = linspace(a,b,N+1);
   h = (b-a)/N;
   y = zeros(N+1,1);
   y(1) = y0;
   q = length(tau);
   for n=1:N
      tn = t(n) + h*tau;
      kn = zeros(q,1);
      g = @(x) f(tn,y(n)+s(x,A,h));
      kn = FPS(kn,g,Nfp);
      y(n+1) = y(n) + h*bhta*kn;
   end
end
```

Για την επίλυση του μη γραμμικού συστήματος για τον υπολογισμό των $k^{n,i}$, η παραπάνω μέθοδος RKI, χρησιμοποιεί τη μέθοδο σταθερού σημείου FPS που δίνεται από τον πιο κάτω κώδικα:

```
function x = FPS(x0, g, Nfp)
  n = length(x0);
  x = x0;

  for i = 1:Nfp
     xold=x;
     for j = 1:n
        gx = g(xold);
        x(j) = gx(j);
     end
  end
end
```

Εφόσον $k^{n,i}=f(t^{n,i}, y^n+s)$, όπου $s = h\sum_{j=1}^{q} a_{ij}k^{n,j}$, η μέθοδος σταθερού σημείου για συστήματα δέχεται ως χ το διάνυσμα k^n και ως συνάρτηση g(x) τη συνάρτηση $f(t^n, y^n+s)$ και επιστρέφει την εκτιμώμενη προσέγγιση για δοσμένο αριθμό επαναλήψεων Nfp.
Για τον υπολογισμό του s χρησιμοποιείται η βοηθητική συνάρτηση:

```
function y = s(x, A, h)
  n = length(x);
  y = zeros(n, 1);
  for i = 1:n
     y(i) = h * A(i, :) * x;
  end
end
```

Στο MATLAB υλοποιήθηκε ο παρακάτω κώδικας για την εφαρμογή της παραπάνω μεθόδου για N=10, 20, 40, 80 και Nfp=2, 7. Για τις προαναφερθέντες τιμές του N και του Nfp υπολογίζονται το μέγιστο απόλυτο σφάλμα, καθώς και η πειραματική τάξη ακρίβειας. Τα σφάλματα αποθηκεύονται στο διάνυσμα «errs» και η πειραματική τάξη ακρίβειας στο διάνυσμα «rates».

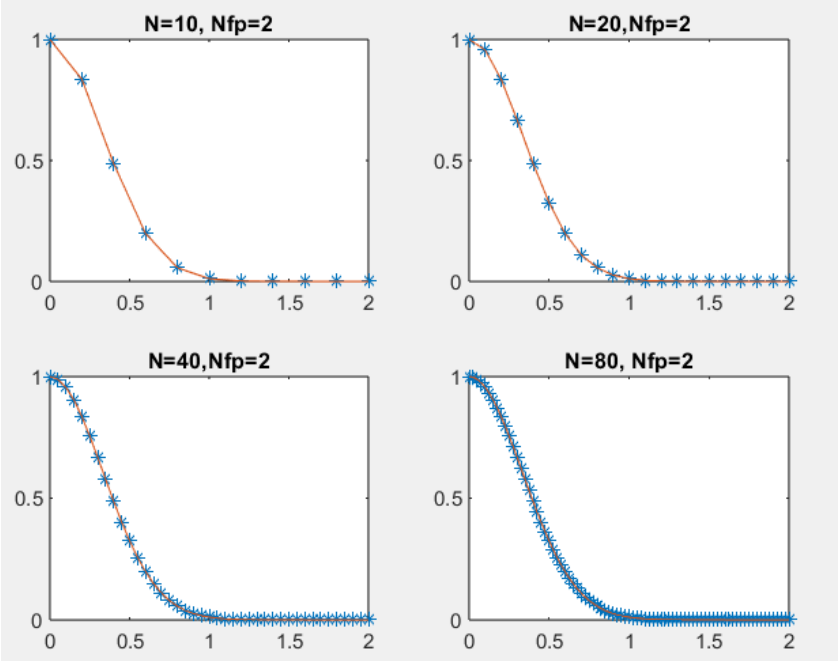
```
clear; clc;
a=0;
b=2;
y0=1;
f=@(t,y) log(y.^2+1)-9*t.*exp(-(9/2)*t.^2)-log(exp(-9*t.^2)+1);
yexact=@(t)exp(-9*t.^2 /2);

A=[5/12 -1/12; 3/4 1/4];
bhta=[3/4 1/4];
tau=[1/3; 1];

Ns=[10, 20, 40, 80];
Nfp=[2 7];
errs=zeros(length(Ns),1);
rates=zeros(length(Ns)-1,1);

for j=1:length(Nfp)
  figure;
  for k=1:length(Ns)
    N=Ns(k);
    [Y, t]= RKI(a,b,y0, N, A, bhta, tau ,f, Nfp(j));
    yex=yexact(t);
    errs(k,j)=max(abs(Y(:)-yex(:)));
    subplot(2,2,k);
    plot(t,Y,'*',t,yex);
    title(["N=" num2str(N) ",Nfp=" num2str(Nfp(j))]);
  end
  for i=1:length(Ns)-1
    rates(i,j)=log(errs(i,j)/errs(i+1,j))/log(Ns(i+1)/Ns(i));
  end
end
errs
rates
```

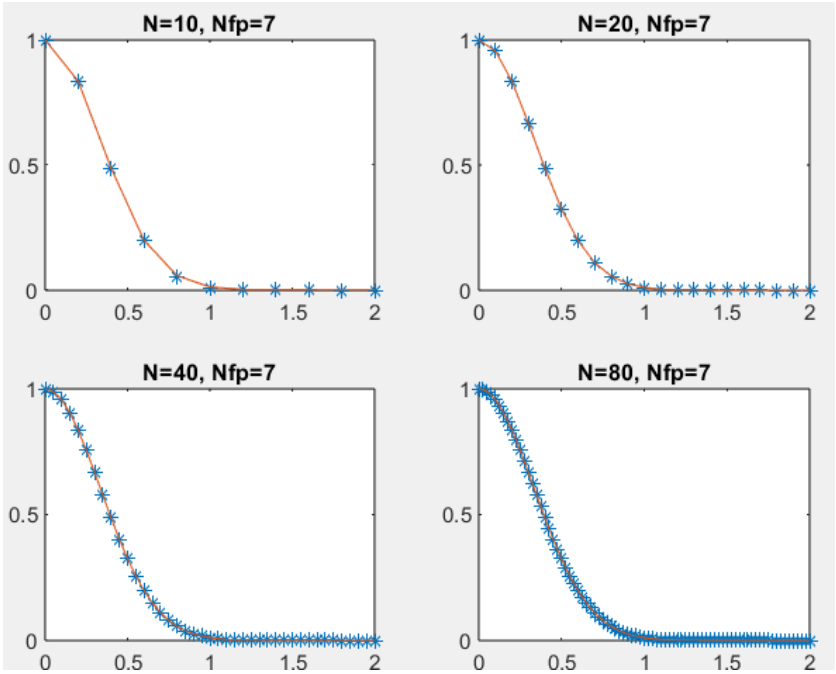
Πιο κάτω παρουσιάζονται τα αποτελέσματα που εμφανίζονται στο Command Window κατά την εκτέλεση του πιο πάνω κώδικα.

```
errs =                    rates =

    0.0018    0.0025          1.0616    3.1261
    0.0009    0.0003          1.6943    3.0535
    0.0003    0.0000          1.8684    3.0234
    0.0001    0.0000
```

Από τα πιο πάνω αποτελέσματα παρατηρούμε ότι για j=1, δηλαδή για Nfp=2, τα σφάλματα είναι αρκετά μικρά, δηλαδή έχουμε καλή προσέγγιση με τη μέθοδο σταθερού σημείου ακόμη και για μικρό αριθμό επαναλήψεων. Επίσης, η πειραματική τάξη ακρίβειας πλησιάζει το 2 για όλα τα N=10,20,40 και 80. Για Nfp=7 (j=2), τα σφάλματα είναι ακόμη μικρότερα και η πειραματική τάξη ακρίβειας τείνει στο 3.

Γραφήματα που προέκυψαν:





Ερώτημα 3

Για την υλοποίηση της πολυβηματικής μεθόδου κατασκευάσαμε τον πιο κάτω κώδικα στο MATLAB:

```
function [y,t]=BDF2ex3(a,b,y0,N,f,Df,maxits)
  h=(b-a)/N; t=linspace(a,b,N+1);
  y=zeros(1,N+1); y(1)=y0;
  t(1)=a;
  t(2)=t(1)+h;
  x0=y(1);
  for i=1:maxits
    xnew = x0 - (x0-y(1) -h/2*(f(t(1) ,y(1))+f(t(2) ,x0)))/(1-h/2*Df(t(2) ,x0));
    x0=xnew;
  end
  y(2)=xnew;
  for n=1:N-1
   Y0=y(n);Y1=y(n+1);Y2=y(n+2);
   for k=1:maxits
     g=Y2-Y0-h/3*(f(t(n+2),Y2)+4*f(t(n+1),Y1)+f(t(n),Y0));
     Dg=1-h/3*Df(t(n+2),Y2);
     Y2=Y2-g/Dg;
   end
   y(n+2)=Y2;
  end
end
```

Στον παραπάνω κώδικα χρησιμοποιείται η έμμεση μέθοδος του τραπεζίου για τον υπολογισμό των κατάλληλων αρχικών συνθηκών.
Στη μέθοδο του τραπεζίου έχουμε ότι:
$Y_{n+1} = Y_n + (h/2) [f(t_n, Y_n) + f(t_{n+1}, Y_{n+1})]$
Θέλουμε να βρούμε το Y_{n+1}. Επομένως θέτουμε $x= Y_{n+1}$

- ⇨ $x= Y_n + (h/2) [f(t_n, Y_n) + f(t_{n+1}, x)]$
- ⇨ $x- Y_n + (h/2) [f(t_n, Y_n) + f(t_{n+1}, x)]=0$
- ⇨ $g(x)=x-Y_n + (h/2) [f(t_n, Y_n) + f(t_{n+1}, x)]$
- ⇨ $g'(x)=1-(h/2)(\partial f/\partial x)$

Συνεπώς, για την Newton-Raphson που υπολογίζει τις κατάλληλες αρχικές συνθήκες στη μέθοδο του τραπεζίου έχουμε:

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} = x_k - \frac{x_k - Y_n - \frac{h}{2}[f(t_n, Y_n) + f(t_{n+1}, x_k)}{1 - \frac{h}{2} f_y(t_{n+1}, x_k)}$$

Για την BDF έχουμε:

$y_{n+2} - y_n = h (1/3\ f_{n+2} + 4/3\ f_{n+1} + 1/3\ f_n)$

⇨ $y_{n+2} - y_n - h (1/3\ f_{n+2} + 4/3\ f_{n+1} + 1/3\ f_n) =0$

Θέτουμε $x= y_{n+2}$

- ⇨ $g(x)=x - y_n - h\,(1/3\, f_{n+2} + 4/3\, f_{n+1} + 1/3\, f_n)$
- ⇨ $g'(x)=1-h/3\, f_y(t_{n+2},x)$

Επομένως, για την NR έχουμε

$$x_{k+1} = x_k - \frac{x_k - y_n - \frac{h}{3}[f(t_{n+2},Y_2) + 4f(t_{n+1},Y_1) + f(t_n,Y_0)}{1 - \frac{h}{3} f_y(t_{n+2},Y_2)}$$

Στο MATLAB υλοποιήθηκε ο παρακάτω κώδικας για την εφαρμογή της παραπάνω μεθόδου για N=20, 40, 80, 160. Για τις προαναφερθέντες τιμές του N υπολογίζονται το μέγιστο απόλυτο σφάλμα, καθώς και η πειραματική τάξη ακρίβειας. Τα σφάλματα αποθηκεύονται στο διάνυσμα «errs» και η πειραματική τάξη ακρίβειας στο διάνυσμα «rates».

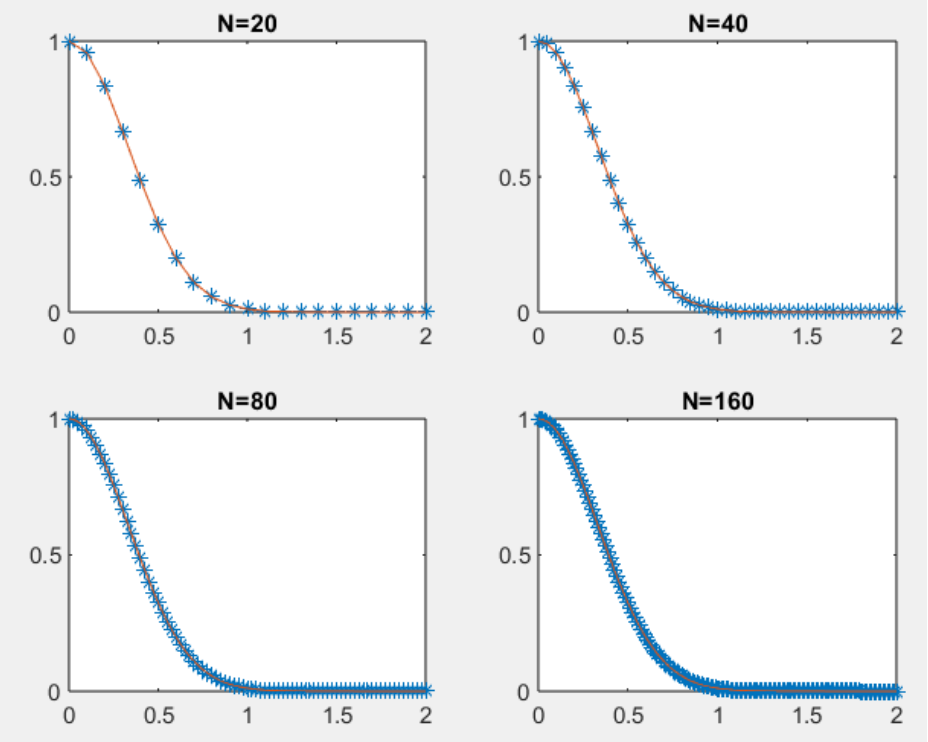
```
clear; clc;
a=0;
b=2;
f=@(t,y) log(y^2+1)-9*t*exp(-(9/2)*t^2)-log(exp(-9*t^2)+1);
yexact=@(t)exp(-9*t.^2 /2);
Df=@(t,y)(2*y)/(y^2+1);
y0=1;maxits=3;
Ns=[20, 40, 80, 160];
errsBDF2=zeros(length(Ns),1); ratesBDF2=zeros(length(Ns)-1,1);
for i=1:length(Ns)
  N=Ns(i);
  [yBDF2,t]=BDF2ex3(a,b,y0,N,f,Df,maxits);
  yex=yexact(t);
  errsBDF2(i)=max(abs(yex(:)-yBDF2(:)));
  subplot(2,2,i);
  plot(t,yBDF2,'*',t,yex);
  title(["N=" num2str(N)]);
end
for i=1:length(Ns)-1
  ratesBDF2(i)=log(errsBDF2(i)/errsBDF2(i+1))/log(Ns(i+1)/Ns(i));
end
errsBDF2
ratesBDF2
```

Πιο κάτω παρουσιάζονται τα αποτελέσματα που εμφανίζονται στο Command Window κατά την εκτέλεση του πιο πάνω κώδικα.

```
errsBDF2 =

    0.0011
    0.0001
    0.0000
    0.0000


ratesBDF2 =

    4.0000
    4.0038
    4.0029
```

Παρατηρούμε ότι οι προσεγγίσεις ήταν αρκετά καλές, ακόμη και για μικρές διαμερίσεις. Η πειραματική τάξη ακρίβειας, όπως φαίνεται, είναι περίπου 4.

Γραφήματα που προέκυψαν:



Ερώτημα 4

Για την υλοποίηση του προβλήματος χρησιμοποιώντας την Runge-Kutta που αντιστοιχεί στο ταμπλό του Butcher του ερωτήματος 1, δημιουργούμε την συνάρτηση RKsys για την επίλυση συστημάτων με Runge-Kutta. Κώδικας:

```
function y= RKsys(a, b, y0, N, A, bhta, tau, f)
  h=(b-a)/N;
  d=length(y0);
  y=zeros(d, N+1);
  y(:, 1)=y0;

  q=length(bhta);
  kn=zeros(d,q);

  t = linspace(a, b, N+1);

  for n = 1:N
    for i = 1:q
      tni = t(n) + tau(i) * h;
      s = zeros(d, 1);
      for j = 1:i-1
        s = s + A(i, j) * kn(:, j);
      end
```

```
        kn(:, i) = f(tni, y(:, n) + h * s);
      end
      y(:, n+1) = y(:,n)+h*kn*bhta;
    end
end
```

Στο MATLAB υλοποιήθηκε ο παρακάτω κώδικας για την εφαρμογή της παραπάνω μεθόδου [%(A)] για h=[0.1, 0.01, 0.001]. Για τις προαναφερθέντες τιμές του h υπολογίζονται οι τιμές των y και t.

```
clear; clc;
a=0.0;
b=2.0;
y0=[1;1];
B=[999, 2024; -1002, -2025];
A=[0 0 0 0; 1/3 0 0 0; -1/3 1 0 0; 1 -1 1 0];
tau=[0; 1/3; 2/3; 1];
bhta=[1/8; 3/8; 3/8; 1/8];
f=@(t,y)B*y;

KNR=3;
h=[0.1, 0.01, 0.001];

for j=1:3
   %(A)
   N=(b-a)/h(j);
   t = linspace(a, b, N+1)
   yRK= RKsys(a, b, y0, N, A, bhta, tau, f)

   figure(1);
   subplot(2,2,j)
   plot( t, yRK);
   title(["h = " num2str(h(j))]);
   legend('y1','y2');

   %(Γ)
   [t,y] = trapezSys(a,b,h(j),y0,f,B,KNR);
   figure(2)
   subplot(2,2,j)
   plot(t,y);
   title(["h = " num2str(h(j))]);
   legend('y1','y2');
end

%(B)
h=0.0001;
[t,y]=trapezSys(a,b,h,y0,f,B,KNR);
figure(3)
plot(t,y);
legend('y1','y2');
```

Πιο κάτω παρουσιάζονται τα αποτελέσματα που εμφανίζονται στο Command Window κατά την εκτέλεση του πιο πάνω κώδικα.

(A) Για h=0.1

```
t =

  Columns 1 through 5

         0    0.1000    0.2000    0.3000    0.4000

  Columns 6 through 10

    0.5000    0.6000    0.7000    0.8000    0.9000

  Columns 11 through 15

    1.0000    1.1000    1.2000    1.3000    1.4000

  Columns 16 through 20

    1.5000    1.6000    1.7000    1.8000    1.9000

  Column 21

    2.0000
```

```
yRK =

  1.0e+133 *

  Columns 1 through 5

    0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000

  Columns 6 through 10

   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000

  Columns 11 through 15

   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000

  Columns 16 through 20

   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000

  Column 21

   -1.8039
    1.8004
```

Για h=0.01

t=

```
 Columns 196 through 200

   1.9500    1.9600    1.9700    1.9800    1.9900

 Column 201

   2.0000
```

yRK=

```
  Columns 196 through 200

       NaN       NaN       NaN       NaN       NaN
       NaN       NaN       NaN       NaN       NaN

  Column 201

       NaN
       NaN
```

Για h=0.001

t=

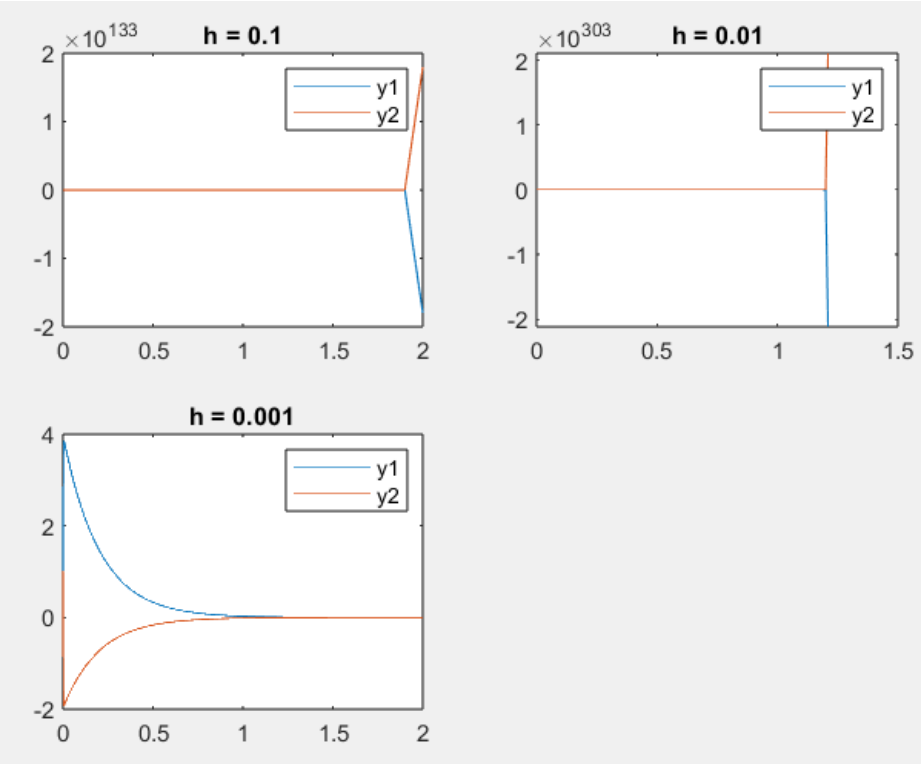
```
 Columns 1,996 through 2,000

   1.9950    1.9960    1.9970    1.9980    1.9990

 Column 2,001

   2.0000
```

yRK=

```
  Columns 1,996 through 2,000

    0.0002    0.0002    0.0002    0.0002    0.0002
   -0.0001   -0.0001   -0.0001   -0.0001   -0.0001

  Column 2,001

    0.0002
   -0.0001
```

Για h=0.01 και h=0.001 δίνονται ενδεικτικά τα τελευταία αποτελέσματα.

Γραφήματα που προέκυψαν:



Παρατηρούμε ότι οι τιμές του y ξεφεύγουν για h=0.1 και αυτό οφείλεται στο ”μεγάλο” μέγεθος του h. Παρόμοιες τιμές λαμβάνουμε και για h=0.01. Ωστόσο, για h=0.001, το γράφημα φαίνεται να ακολουθεί μία «λογική» συνάρτηση.

(Β) Για να λυθεί το πρόβλημα με την έμμεση μέθοδο του τραπεζίου, κατασκευάστηκε η συνάρτηση trapezSys, η οποία υλοποιεί τη μέθοδο του τραπεζίου για συστήματα. Για την επίλυση του μη γραμμικού συστήματος σε κάθε επανάληψη της μεθόδου του τραπεζίου χρησιμοποιήθηκε η μέθοδος Newton-Raphson για συστήματα, η οποία υλοποιήθηκε με τη συνάρτηση trapSysNR. Αν $F(t, f) = (f'_1, f'_2) = (999f_1 + 2024f_2, -1002f_1 - 2025f_2)$, σύμφωνα με το δοσμένο σύστημα, τότε για τη μέθοδο του τραπεζίου έχουμε $Y_{n+1} = Y_n + h/2\,(F_n + F_{n+1})$, οπότε η συνάρτη G στη Newton-Raphson θα είναι η $G(x) = x - Y_n - h/2\,[F(t_n, Y_n) + F(t_{n+1}, x)]$. Επομένως ο Ιακωβιανός πίνακας της G θα είναι $JG(x) = I - h/2\,JF(t_{n+1}, x)$ και ο Ιακωβιανός πίνακας της Φ είναι προφανώς πίνακας σταθερών B= [999 2024; -1002 -2025]. Επομένως, σε κάθε επανάληψη της μεθόδου του τραπεζίου, η Newton-Raphson που θα υπολογίζει σε KNR βήματα την επόμενη προσέγγιση θα είναι η ακόλουθη:

$$x_{n+1}^{(0)} = x_n$$

$$x_{n+1}^{(k+1)} = x_{n+1}^{(k)} - \left[JG\left(x_{n+1}^{(k)}\right)\right]^{-1} G(x_{n+1}^{(k)})$$

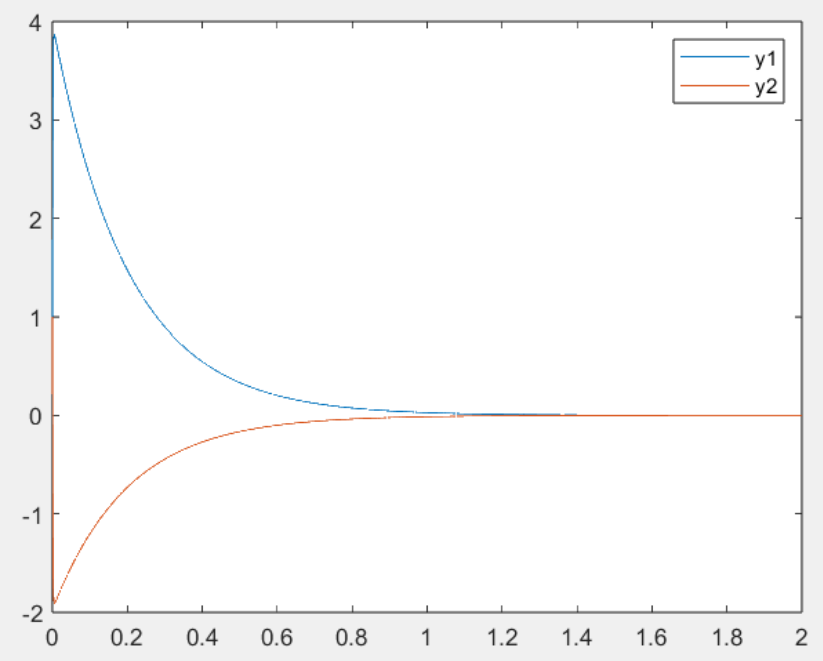
Αν ο Ιακωβιανός πίνακας της G δεν ήταν πίνακας σταθερών και είχαμε και μεγαλύτερο σύστημα, για την αποφυγή υπολογισμού του αντιστρόφου σε κάθε βήμα θα υπολογίζαμε:

$$JG\left(x_{n+1}^{(k)}\right) x_{n+1}^{(k+1)} = JG\left(x_{n+1}^{(k)}\right) x_{n+1}^{(k)} - G(x_{n+1}^{(k)})$$

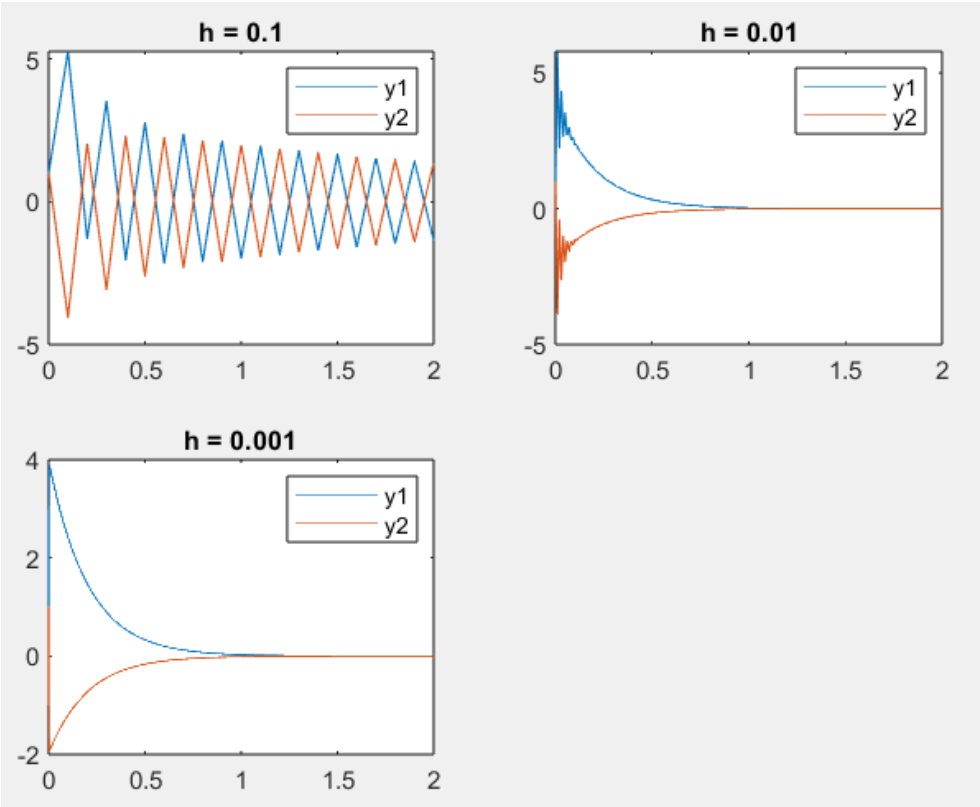
Κώδικας:

```
function [t,Y] = trapezSys(a,b,h,Y0,F,JF,KNR)
   N = (b-a)/h;
   t = linspace(a,b,N+1);
   Y = zeros(length(Y0),N+1);
   Y(:,1) = Y0;
   for i=1:N
   Y(:,i+1) = trapSysNR(t(i+1),Y(:,i),h,F,JF,KNR);
   end
end

function sol = trapSysNR(tnew ,Yold ,h,F,JF,KNR)
   x0 = Yold;
   told = tnew -h;
   G = @(x) x-x0-h/2*(F(told ,x0)+F(tnew ,x));

   for k=1:KNR
   JG = eye(length(Yold))-h/2*JF;
   sol = JG\(JG*Yold -G(Yold));
   Yold = sol;
   end
end
```

Στο MATLAB υλοποιήθηκε κώδικας που φαίνεται στις προηγούμενες σελίδες [%(B)] για την εφαρμογή της παραπάνω μεθόδου για h=0.0001. Για τη συγκεκριμένη τιμή του h επιλύθηκε το σύστημα και πήραμε το ακόλουθο γράφημα το οποίο έχει μεγάλη ομοιότητα με το 3ο γράφημα που προέκυψε στο ερώτημα (A) για h=0.001.



(Γ) Για τη λύση του συστήματος με την έμμεση μέθοδο του τραπεζίου για h=0.1, h=0.01 και h=0.001 χρησιμοποιήθηκε ο κώδικας που φαίνεται στις προηγούμενες σελίδες [%(Γ)] και πήραμε τα πιο κάτω γραφήματα:



Από τα παραπάνω γραφήματα παρατηρούμε ότι για h=0.001 δεν φαίνεται να διαφέρει η λύση από την πραγματική. Επίσης, για h=0.01 αρχικά οι λύσεις τις μεθόδου ταλαντώνονται και στη συνέχεια εξομαλύνονται και παίρνουν την συμπεριφορά της πραγματικής λύσης. Ενώ για h=0.1 οι λύσεις ταλαντώνονται διαρκώς και δεν μοιάζουν καθόλου με την πραγματική λύση.

Επομένως, από όλα τα παραπάνω συμπεραίνουμε πως η έμμεση μέθοδος του τραπεζίου είναι πιο αποτελεσματική για μικρές διαμερίσεις. Επίσης, από τα αποτελέσματα και τις γραφικές παραστάσεις παρατηρούμε πως η μέθοδος του τραπεζίου δίνει καλύτερα αποτελέσματα από την Runge-Kutta.

Ερώτημα 5

Για την υλοποίηση της άσκησης κατασκευάζουμε στη MATLAB την συνάρτηση FDM_Dirichlet, η οποία με βάση τα όρια του διαστήματος [a,b], τις συνοριακές συνθήκες u(a)=A και u(b)=B και τις συναρτήσεις r και f κατασκευάζει τον πίνακα M και το διάνυσμα F και λύνει το σύστημα My=F. Κώδικας:

```
function [U]= FDM_Dirichlet(a, b, A, B, r, f, N)
  h=(b-a)/N;
  x = linspace(a, b, N+1);
  U=zeros(1, N+1);

  U(1)=A;
  U(N+1)=B;
```

```
  a1=(-1/h^2)*ones(N-1,1);
  a2=(2/h^2)*ones(N-1,1)+r(x(2:N))';
  a3=(-1/h^2)*ones(N-1,1);

  M=spdiags([a1, a2, a3], [-1, 0, 1], N-1, N-1);
  condest(M)

  F= f(x(2:N))';
  F(1) = f(x(2)) + A / h^2;
  F(N-1) = f(x(N)) + B / h^2;

  Uint=M\F;
  U(2:N)=Uint';
end
```

Το δοσμένο πρόβλημα δεν είναι στη μορφή που το θέλουμε επομένως πολλαπλασιάζουμε και τα 2 μέλη της διαφορικής εξίσωσης με $r=10^k$:

$-10^{-k}\, u''(x) + u(x) = [-10^{-k}(4x^2 - 2) + 1] \exp(-x^2+1)$, $0 < x < 1$, $u(0) = e$, $u(1) = 1$

⇨ $-u''(x) + 10^k u(x) = [-(4x^2 - 2) + 10^k] \exp(-x^2+1)$, $u(0) = e$, $u(1) = 1$

Επομένως ορίζουμε $r(x)= 10^k$ και $f(x)= [-(4x^2 - 2) + r(x)] \exp(-x^2+1)$.

Στο MATLAB υλοποιούμε τον πιο κάτω κώδικα που κάνει χρήση της πιο πάνω συνάρτηση για την υλοποίηση της άσκησης.

```
clear;clc;
a=0;
b = 1;
h = [0.1 0.001];
k = [-2 -1 0 1 2];
A = exp(1);
B = 1;
for m=1:length(h)
  Ns(m)=(b-a)/h(m);
end
uexact = @(x) exp(1-x.^2);
errs=zeros(length(Ns),length(k));
rates=zeros(length(Ns)-1, length(k));
figure;
for m=1:length(Ns)
  for j=1:length(k)
    r = @(x) 10.^(k(j));
    f = @(x) exp((-x.^2)+1).*(-(4*x.^2-2)+r(x));
    disp(["Deiktis katastasis gia h=" h(m), "k=" num2str(k(j))]);
    U= FDM_Dirichlet(a, b, A, B, r, f, Ns(m));
    x=linspace(a, b, Ns(m) + 1);
    errs(m,j)=max(abs(uexact(x)-U));
    subplot(length(h), length(k), (m - 1) * length(k) + j)
    plot(x,U,'*',x,uexact(x));
```

```
    title(sprintf('h = %.3f, k = %d', h(m), k(j)));
  end
end
for j=1:length(k)
  for i=1:length(Ns)-1
    rates(i,j)=log(errs(i,j)/errs(i+1,j))/log(Ns(i+1)/Ns(i));
  end
end
errs
rates
```

Πιο κάτω παρουσιάζονται τα αποτελέσματα που εμφανίζονται στο Command Window κατά την εκτέλεση του πιο πάνω κώδικα.

```
    "Deiktis katastasi..."    "0.1"    "k="    "-2"

ans =

   49.9488

    "Deiktis katastasi..."    "0.1"    "k="    "-1"

ans =

   49.4927

    "Deiktis katastasi..."    "0.1"    "k="    "0"

ans =

   45.3514

    "Deiktis katastasi..."    "0.1"    "k="    "1"

ans =

   24.7173

    "Deiktis katastasi..."    "0.1"    "k="    "2"

ans =

    4.9187

    "Deiktis katastasi..."    "0.001"    "k="    "-2"

ans =

   4.9948e+05

    "Deiktis katastasi..."    "0.001"    "k="    "-1"

ans =

   4.9484e+05

    "Deiktis katastasi..."    "0.001"    "k="    "0"

ans =

   4.5272e+05
```

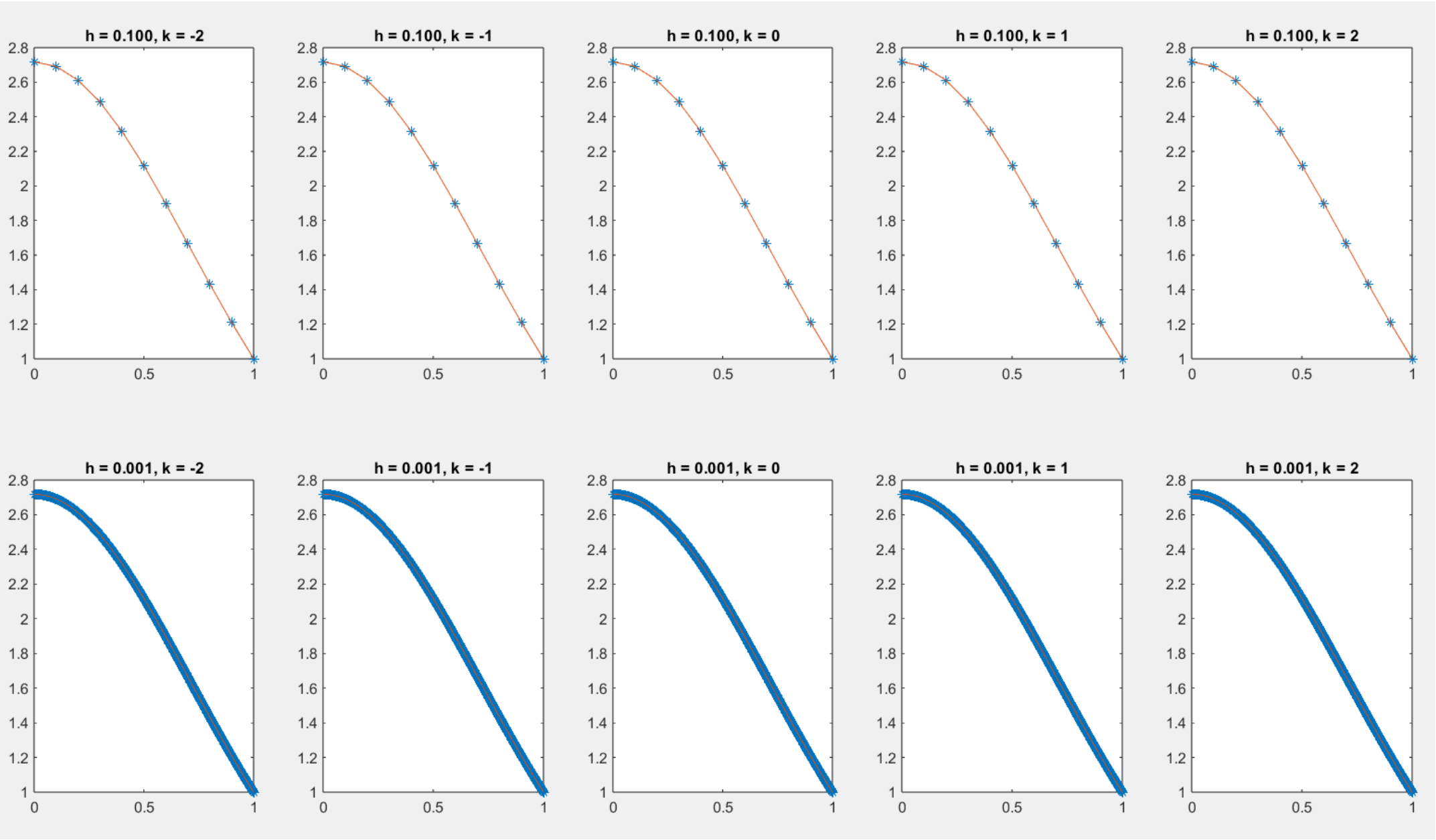
```
    "Deiktis katastasi..."    "0.001"    "k="    "1"

ans =

   2.4209e+05

    "Deiktis katastasi..."    "0.001"    "k="    "2"

ans =

   3.9462e+04
```

Από τα παραπάνω αποτελέσματα του δείκτη κατάστασης βλέπουμε πως όσο μεγαλώνει το k για σταθερό h, τόσο μικραίνει ο δείκτης κατάστασης του πίνακα M. Επιπλέον, όσο μεγαλώνει η διάσταση του πίνακα M, δηλαδή μικραίνει το h, για σταθερό k, τόσο μεγαλώνει ο δείκτης κατάστασης.

```
errs =

   1.0e-03 *

    0.7246    0.7209    0.6866    0.5090    0.1657
    0.0001    0.0001    0.0001    0.0001    0.0000


rates =

    1.9947    1.9946    1.9935    1.9960    1.9949
```

Από τα παραπάνω αποτελέσματα παρατηρούμε πως όσο μειώνεται ο δείκτης κατάστασης για σταθερή διάσταση του πίνακα μειώνονται και τα μέγιστα απόλυτα σφάλματα. Επίσης, από τις τιμές του διανύσματος “rates” βλέπουμε πως η πειραματική τάξη ακρίβειας τείνει στο 2.

Στην επόμενη σελίδα παρουσιάζονται τα διαγράμματα για όλες τις πιο πάνω περιπτώσεις που εξετάστηκαν.


h = 0.100, k = -2
h = 0.100, k = -1
h = 0.100, k = 0
h = 0.100, k = 1
h = 0.100, k = 2
h = 0.001, k = -2
h = 0.001, k = -1
h = 0.001, k = 0
h = 0.001, k = 1
h = 0.001, k = 2
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0
0.5
1