

### 3ο Εργαστήριο

Για την αριθμητική επίλυση ενός προβλήματος αρχικών τιμών (Π.Α.Τ.)

$$y'(t) = f(t, y(t)), \quad t \in [a, b], \quad y(0) = y_0$$

θεωρήσαμε άμεσες μεθόδους όπως του Euler αλλά και πεπλεγμένες μεθόδους όπως του Euler και του τραπεζίου. Έστω ένας ομοιόμορφος διαμερισμός του  $[a, b]$ , στα σημεία  $t_n = a + nh$ ,  $n = 0, \dots, N$ , με βήμα  $h = \frac{b-a}{N}$ , υπολογίζουμε τις τιμές  $y_n$  που αποτελούν προσεγγίσεις στις τιμές  $y(t_n)$ ,  $n = 0, \dots, N$ .

#### Άμεση Euler

$$y_{n+1} = y_n + hf(t_n, y_n), \quad n = 0, \dots, N-1.$$

#### Πεπλεγμένη Euler

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad n = 0, \dots, N-1.$$

#### Μέθοδος του τραπεζίου

$$y_{n+1} = y_n + \frac{h}{2}(f(t_{n+1}, y_{n+1}) + f(t_n, y_n)), \quad n = 0, \dots, N-1.$$

Για να μπορέσουμε να βρούμε την προσέγγιση αν θεωρήσουμε μια πεπλεγμένη μέθοδο θα χρειαστεί να βρούμε τη ρίζα μιας μη-γραμμικής συνάρτησης σε κάθε βήμα. Αν όμως η συνάρτηση  $f$  είναι γραμμική ως προς  $y$ , π.χ.  $f(t, y) = g(t)y$ , τότε οι παραπάνω πεπλεγμένες μέθοδοι υλοποιούνται εύκολα, π.χ.

#### Πεπλεγμένη Euler

$$y_{n+1} = \frac{1}{1 - hg(t_{n+1})} y_n, \quad n = 0, \dots, N-1.$$

#### η Μέθοδος του τραπεζίου

$$y_{n+1} = \frac{1 + h\frac{g(t_n)}{2}}{1 - h\frac{g(t_{n+1})}{2}} y_n, \quad n = 0, \dots, N-1.$$

**Άσκηση 1:** Έστω  $y(t) = e^{0.25-(t-0.5)^2}$ , στο  $[0, 4]$  η οποία είναι λύση στο

$$y'(t) = (1 - 2t)y(t), \quad t \in [0, 4], \quad y(0) = 1.$$

Θεωρείστε την άμεση μέθοδο του Euler, την πεπλεγμένη μέθοδο του Euler και τη μέθοδο τραπεζίου.

Θεωρήστε ένα διαμερισμό του  $[0, 4]$  σε  $N + 1$  σημεία. Για  $N = 50$ , κατασκευάστε τις προσεγγίσεις που δίνουν οι τρεις μέθοδοι, δημιουργήστε τις γραφικές παράστασεις της προσεγγιστικής λύσης και της ακριβούς στο διάστημα  $[0, 4]$ . Στη συνέχεια βρείτε το σφάλμα  $\max_{0 \leq n \leq N} |y_n - y(t_n)|$ . Για  $N = 100, 200, 300, 400, 500$  βρείτε τη πειραματική τάξη σύγκλισης για κάθε μια από τις τρεις μεθόδους

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
def y_exact(t):
    s=np.exp(0.25-(t-0.5)**2)
    return s

def f(t,y):
    s=(1-2*t)*y
    return s

def g(t):
    s=(1-2*t)
    return s

####Euler
N=50
t=np.linspace(0,4,N+1)
h=t[1]-t[0]

y=np.zeros(N+1)
y[0]=1

for i in range(N):
    y[i+1]=y[i]+h*f(t[i],y[i])

plt.plot(t,y_exact(t),t,y)
plt.show()
#e4=max(abs(y_exact(t)-y))
#print(e4)

##### Πεπλεγμένη Euler
for i in range(N):

    y[i+1]=y[i]/(1-h*g(t[i+1]))

e=max(abs(y_exact(t)-y))
plt.plot(t,y_exact(t),t,y)
plt.show()

## Τραπεζίου
for i in range(N):

    y[i+1]=y[i]*(1+h*g(t[i])/2)/(1-h*g(t[i+1])/2)

e=max(abs(y_exact(t)-y))
plt.plot(t,y_exact(t),t,y)
plt.show()

#####

err=np.zeros(5) #array για τα σφάλματα
N=[100,200,300,400,500] #διαμερίσεις
N_len=len(N) #μήκος N
for j in range(N_len):
```

```

y=np.zeros(N[j]+1)
y[0]=1

t=np.linspace(0,4,N[j]+1)
h=t[1]-t[0]

for i in range(N[j]):
    y[i+1]=y[i]+h*f(t[i],y[i])

err[j]=max(abs(y_exact(t)-y)) #σφάλμα
print('Μέγιστο σφάλμα για N=',N[j],':',err[j])

print('Πειραματική ταξη σύγκλισης μεθόδου Euler')
for i in range(N_len-1):
    p=np.log(err[i+1]/err[i])/np.log(N[i]/N[i+1])
    print(p)

### Πεπλεγμένη Euler
err=np.zeros(5) #array για τα σφάλματα
N=[100,200,300,400,500] #διαμερίσεις
N_len=len(N) #μήκος N
for j in range(N_len):

    y=np.zeros(N[j]+1)
    y[0]=1

    t=np.linspace(0,4,N[j]+1)
    h=t[1]-t[0]

    for i in range(N[j]):
        y[i+1]=y[i]/(1-h*g(t[i+1]))

    err[j]=max(abs(y_exact(t)-y)) #σφάλμα
    print('Μέγιστο σφάλμα για N=',N[j],':',err[j])

print('Πειραματική ταξη σύγκλισης μεθόδου Πεπλεγμένη Euler')
for i in range(N_len-1):
    p=np.log(err[i+1]/err[i])/np.log(N[i]/N[i+1])
    print(p)

### Τραπεζίον
err=np.zeros(5) #array για τα σφάλματα
N=[100,200,300,400,500] #διαμερίσεις
N_len=len(N) #μήκος N
for j in range(N_len):

    y=np.zeros(N[j]+1)
    y[0]=1

    t=np.linspace(0,4,N[j]+1)
    h=t[1]-t[0]

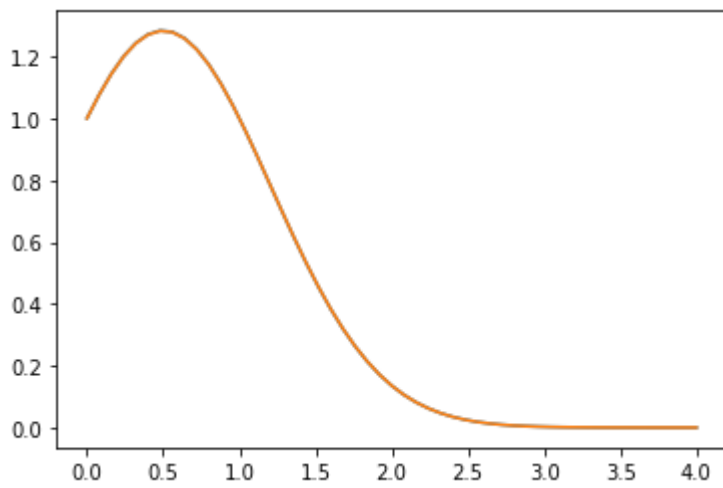
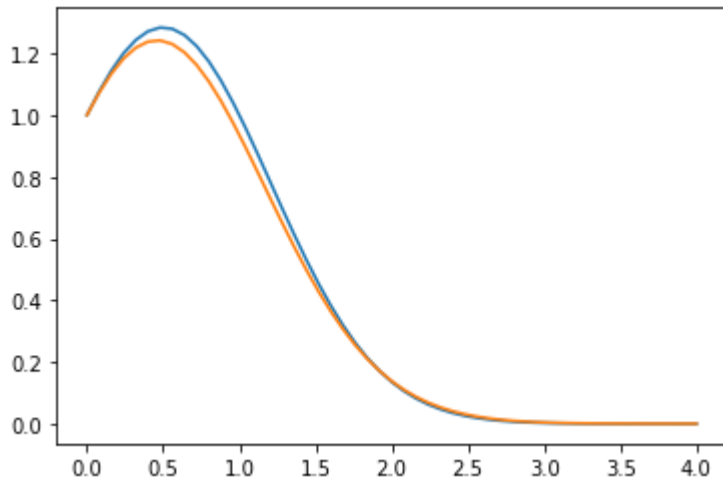
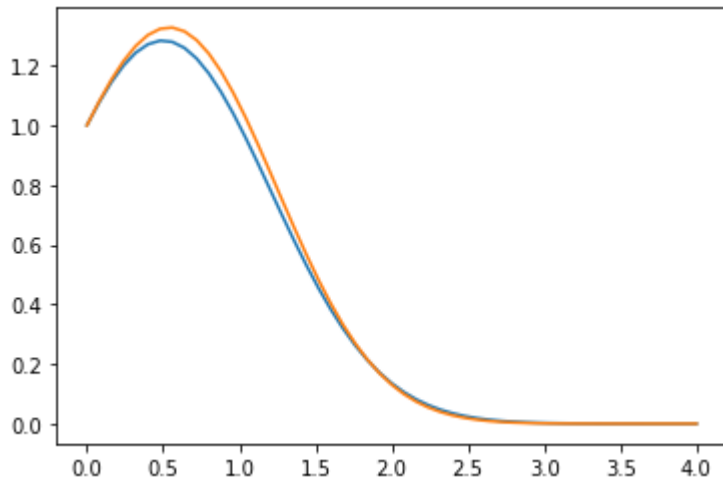
    for i in range(N[j]):
        y[i+1]=y[i]*(1+h*g(t[i])/2)/(1-h*g(t[i+1])/2)

    err[j]=max(abs(y_exact(t)-y)) #σφάλμα
    print('Μέγιστο σφάλμα για N=',N[j],':',err[j])

print('Πειραματική ταξη σύγκλισης μεθόδου Τραπεζίου')

```

```
for i in range(N_len-1):  
    p=np.log(err[i+1]/err[i])/np.log(N[i]/N[i+1])  
    print(p)
```



Μέγιστο σφάλμα για  $N= 100$  : 0.034336411250643684  
 Μέγιστο σφάλμα για  $N= 200$  : 0.01705849284890526  
 Μέγιστο σφάλμα για  $N= 300$  : 0.011347840786325891  
 Μέγιστο σφάλμα για  $N= 400$  : 0.008501683420745643  
 Μέγιστο σφάλμα για  $N= 500$  : 0.006796928524881007  
 Πειραματική ταξη σύγκλισης μεθόδου Euler  
 1.009249073246151  
 1.0053163616951728  
 1.0037583881814047  
 1.002912113953019  
 Μέγιστο σφάλμα για  $N= 100$  : 0.03345077565804466  
 Μέγιστο σφάλμα για  $N= 200$  : 0.016837019545659082  
 Μέγιστο σφάλμα για  $N= 300$  : 0.01124966549226869  
 Μέγιστο σφάλμα για  $N= 400$  : 0.008446801039597274  
 Μέγιστο σφάλμα για  $N= 500$  : 0.0067619350308429915  
 Πειραματική ταξη σύγκλισης μεθόδου Πεπλεγμένη Euler  
 0.9904028864140126  
 0.994516184744095  
 0.9960669333679791  
 0.9970205140387308  
 Μέγιστο σφάλμα για  $N= 100$  : 0.00023478955730626971  
 Μέγιστο σφάλμα για  $N= 200$  : 5.884425177127284e-05  
 Μέγιστο σφάλμα για  $N= 300$  : 2.6148473388110105e-05  
 Μέγιστο σφάλμα για  $N= 400$  : 1.4712359181379142e-05  
 Μέγιστο σφάλμα για  $N= 500$  : 9.415201378093485e-06  
 Πειραματική ταξη σύγκλισης μεθόδου Τραπεζίου  
 1.9963948464740422  
 2.000426983764215  
 1.9990919270159062  
 2.0003372161207764

## Μη γραμμική $f$ ως προς $y$

Σε αυτή την περίπτωση για να υπολογίσουμε τις προσεγγίσεις χρησιμοποιώντας μια πεπλεγμένη μέθοδο πρέπει να βρούμε τη λύση μιας μη γραμμικής εξίσωσης σε κάθε βήμα. Μια μέθοδος είναι π.χ. η μέθοδος του σταθερού σημείου.

### Μέθοδος σταθερού σημείου

Για να βρούμε τη λύση της εξίσωσης  $x^* = g(x^*)$ , για  $x_0$  δοσμένο δημιουργούμε την ακολουθία  $x_k$

$$x_{k+1} = g(x_k), n = 0, \dots$$

Αν  $x_k$  συγκλίνει τότε  $x_k \rightarrow x^*$ . Επειδή δεν μπορούμε να βρούμε ακριβώς το  $x^*$  αλλά μια προσέγγιση, πρέπει ο αλγόριθμος να σταματά μετά από κάποια βήματα. Ένα κριτήριο τερματισμού του αλγορίθμου είναι να θέσουμε ότι η διαφορά δυο συνεχόμενων προσεγγίσεων  $x_k$  να γίνει μικρότερη από έναν προκαθορισμένο αριθμό TOL. Έτσι τερματίζουμε την επαναληπτική διαδικασία και δεχόμαστε ως προσέγγιση της  $x^*$  το  $x_k$  όταν  $|x_k - x_{k-1}| \leq TOL$ . Επειδή μπορεί να δημιουργηθούν και άλλα σφάλματα και το προηγούμενο κριτήριο να μην ικανοποιείται ποτέ, θέτουμε και για ασφάλεια ένα μέγιστο αριθμό επαναλήψεων  $N_{max}$ . Δηλαδή δεχόμαστε ως  $x^*$  είτε το  $x_k$  που πρώτη φορά ικανοποιείται το κριτήριο τερματισμού ή το  $x_{N_{max}}$ .

**Άσκηση 2 (για την εύρεση ενός σταθερού σημεί: Έστω  $g(t) = \cos(t)$ , στο  $[0, 1]$ . Θέλουμε να βρούμε το σημείο  $x^* \in [0, 1]$  τέτοιο ώστε**

$$x^* = \cos(x^*).$$

Φτιάξτε μια επαναληπτική διαδικασία ώστε να προσεγγίσετε το  $x^*$ , ξεκινώντας από  $x_0 = 1$ . Για κριτήρια τερματισμού, θέσετε μέγιστο αριθμό επαναλήψεων  $Nmax = 500$  και  $TOL = 10^{-8}$ .

In [2]:

```
def g(x):
    s=np.cos(x)
    return s

x0=1 #αρχική προσέγγιση

tol=1.e-8
Nmax=500
k=0

err=1. #Θετουμε αρχικά το σφάλμα ίσον με 1 για να ξεκινήσει η διαδικασία

while (err>tol) and (k<=Nmax):
    x=g(x0) # επόμενη προσέγγιση
    err=abs(x-x0) #σφάλμα

    k=k+1 # αυξάνουμε τον μετρητή βημάτων
    x0=x # θέτουμε τη νέα προσέγγιση ως την παλιά για το επόμενο βήμα

print('Η προσέγγιση του σταθερου σημείου είναι:', x)
print('Αριθμός βημάτων:', k)
```

Η προσέγγιση του σταθερου σημείου είναι: 0.7390851366465718

Αριθμός βημάτων: 46

## Πεπλεγμένη Euler

Μια πεπλεγμένη μέθοδος είναι η **πεπλεγμένη Euler**

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad n = 0, \dots, N - 1.$$

Για να υπολογίσουμε την τιμή  $y_{n+1}$  χρειάζεται να λύσουμε (στη γενική περίπτωση που η  $f$  είναι μη γραμμική ως προς  $y$ ) μια μη γραμμική εξίσωση. Ένας τρόπος είναι χρησιμοποιήσουμε μια επαναληπτική διαδικασία όπως στον υπολογισμό του σταθερού σημείου. Έτσι, σε κάθε βήμα του αλγορίθμου της πεπλεγμένης Euler, για τον υπολογισμό δηλαδή του  $y_{n+1}$  θέτουμε

$$g(s) = y_n + hf(t_{n+1}, s),$$

και αναζητούμε  $s^*$  τέτοιο ώστε

$$s^* = g(s^*).$$

Επειδή μπορούμε μόνο να προσεγγίσουμε το  $s^*$ , την προσέγγιση  $s_k$  που θα υπολογίσουμε χρησιμοποιώντας την επαναληπτική διαδικασία του σταθερού σημείου, θα θέσουμε ως  $y_{n+1}$ , δηλαδή θεωρούμε την ακολουθία

$$s_{k+1} = g(s_k), k = 0, \dots,$$

για δοσμένο tol, υπολογίζουμε το  $s_k$  τέτοιο ώστε  $|s_k - s_{k-1}| \leq tol$  και θέτουμε

$$y_{n+1} = s_k$$

και προχωράμε για την επόμενη προσέγγιση της μεθόδου πεπεπλεγμένης Euler.

</p>

**Άσκηση 3:** θεωρείστε την πεπλεγμένη μέθοδο του Euler για τον υπολογισμό της λύσης του προβλήματος

$$y'(t) = \frac{1}{1+t^2} - 2(y(t))^2, \quad t \in [0, 1], \quad y(0) = 0.$$

Η ακριβή λύση του προβλήματος είναι  $y(t) = \frac{t}{1+t^2}$ . Βρείτε το σφάλμα  $\max_{0 \leq n \leq N} |y_n - y(t_n)|$ . Για  $N = 100, 200, 300, 400, 500$  βρείτε τη πειραματική τάξη σύγκλισης για την πεπλεγμένη μέθοδο του Euler. Θεωρείστε για κάθε βήμα, μέγιστο αριθμό επαναλήψεων  $N_{max} = 500$  και  $TOL = 10^{-8}$



In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
def y_exact(t):
    s=t/(1+t**2)
    return s

def f(t,y):
    s=1/(1+t**2)-2*y**2
    return s

def g(t,yn,x):
    s=yn+h*f(t,x)
    return s

#### Πεπλεγμένη Euler (Μη γραμμική f ως προς y)
N=50
t=np.linspace(0,1,N+1)
h=t[1]-t[0]

y=np.zeros(N+1)
y[0]=0

tol=1.e-8
Nmax=500

for i in range(N):

    x0=y[i] #αρχική προσέγγιση στο i-βημα
    k=0

    err=1. #Θετουμε αρχικά το σφάλμα ίσον με 1 για να ξεκινήσει η διαδικασία

    while (err>tol) and (k<=Nmax):
        x=g(t[i+1],y[i],x0) # επόμενη προσέγγιση
        e=abs(x-x0) #σφάλμα

        k=k+1 # αυξάνουμε τον μετρητή βημάτων
        x0=x

    y[i+1]=x #τελειώνει η επαναληψη σταθερου σημειου και θέτουμε τη προσεγγιση που βρηκα
με
    ##### ως την προσεγγιση της λύσης στο σημείο t[i+1]

plt.plot(t,y_exact(t),t,y)
plt.show()

err_y=np.zeros(5) #array για τα σφάλματα
N=[100,200,300,400,500] #διαμερίσεις
N_len=len(N) #μήκος N

tol=1.e-8
Nmax=500

for j in range(N_len):
```

```

y=np.zeros(N[j]+1)
y[0]=0

t=np.linspace(0,1,N[j]+1)
h=t[1]-t[0]

for i in range(N[j]):
    x0=y[i] #αρχική προσέγγιση στο i-βήμα
    k=0

    err=1. #Θετουμε αρχικά το σφάλμα ίσον με 1 για να ξεκινήσει η διαδικασία

    while (err>tol) and (k<=Nmax):
        x=g(t[i+1],y[i],x0) # επόμενη προσέγγιση
        e=abs(x-x0) #σφάλμα

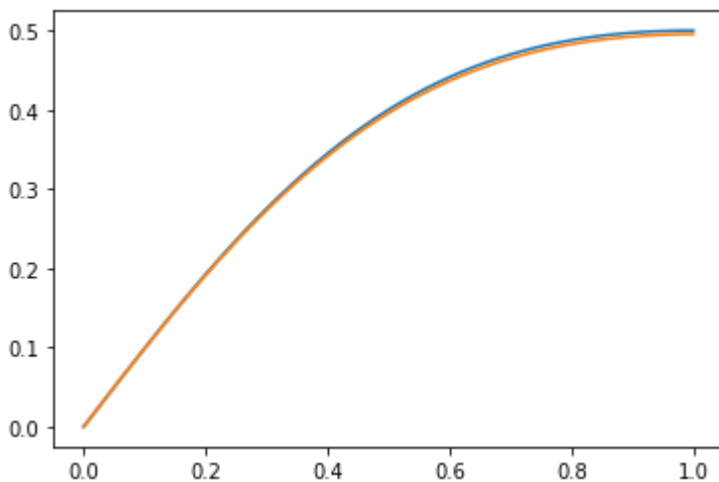
        k=k+1 # αυξάνουμε τον μετροητή βημάτων
        x0=x

    y[i+1]=x #τελειώνει η επαναληψη σταθερου σημειου και θέτουμε τη προσεγγιση που βρη
καμε
    ##### ως την προσεγγιση της λύσης στο σημείο t[i+1]

    err_y[j]=max(abs(y_exact(t)-y)) #σφάλμα
    print('Μέγιστο σφάλμα για N=',N[j],':',err_y[j])

print('Πειραματική τάξη σύγκλισης μεθόδου Πεπλεγμένη Euler')
for i in range(N_len-1):
    p=np.log(err_y[i+1]/err_y[i])/np.log(N[i]/N[i+1])
    print(p)

```



```

Μέγιστο σφάλμα για N= 100 : 0.0024947012134985957
Μέγιστο σφάλμα για N= 200 : 0.0012496582121343525
Μέγιστο σφάλμα για N= 300 : 0.0008336221343824834
Μέγιστο σφάλμα για N= 400 : 0.0006254108958987925
Μέγιστο σφάλμα για N= 500 : 0.0005004230469904392
Πειραματική τάξη σύγκλισης μεθόδου Πεπλεγμένη Euler
0.9973334721384007
0.9984709685506926
0.9989199319921784
0.999155167959594

```

**Άσκηση 4:** Επαναλάβετε το πρόβλημα για τη μέθοδο του τραπεζίου.

In [4]:

```
##### Τραπεζίου (Μη γραμμική f ως προς y)
# Η αλλαγή σε σχέση με την προηγούμενη άσκηση είναι ο ορισμός της συνάρτησης που δίνει τη
# μέθοδο.
# Ο τύπος της μεθόδου του τραπεζίου είναι
#  $y_{n+1} = y_n + (h/2)(f(t_{n+1}, y_{n+1}) + f(t_n, y_n))$ ,

def g(t, yn, x):
    s = yn + (h/2) * (f(t-h, yn) + f(t, x))
    return s
```