Αριθμητική Επίλυση Συνήθων Διαφορικών Εξισώσεων

3ο Εργαστήριο

27/03/2015

Σκοπός αυτού του εργαστηρίου είναι να δούμε πως μπορούμε να επιλύσουμε συστήματα διαφορικών εξισώσεων, με την χρήση του Matlab.

1 Συστήματα διαφορικών εξισώσεων 1ης τάξης

Ένα σύστημα Σ.Δ.Ε. γράφεται ως εξής: Έστω $m \in \mathbb{N}$, $f: [a,b] \times \mathbb{R}^m \to \mathbb{R}^m$ και $y_0 \in \mathbb{R}^m$. Ζητείται συνάρτηση $y : [a,b] \to \mathbb{R}^m$ που να ικανοποιεί

$$\begin{cases} y'(t) = f(t, y(t)), \quad t \in [a,b], \\ y(a) = y_0. \end{cases} \tag{1}$$

Ένα τέτοιο σύστημα λύνεται με αντίστοιχο τρόπο όπως στην περίπτωση των βαθμωτών συναρτήσεων, οπότε η μέθοδος Euler σε αυτή τη περίπτωση γράφεται:

$$y_{(k)}^{n+1} = y_{(k)}^n + hf\left(t^n, y_{(k)}^n\right), \quad n = 1, 2, \ldots, N, \quad k = 1, 2, \ldots, m, \tag{2}$$

όπου $h = (b-a)/N$ είναι το βήμα μας και $y_{(k)}^n$ είναι η k-οστή συνιστώσα της προσέγγισής μας στην χρονική στιγμή t^n, όπου $t^n = t^0 + nh$.
Τέτοια συστήματα προκύπτουν πολλές φορές όταν προσπαθούμε να λύσουμε μια Σ.Δ.Ε. υψηλότερης τάξης την οποία γράφουμε ισοδύναμα ως ένα σύστημα 1ης τάξης. Ένα χαρακτηριστηκό παράδειγμα είναι το απλό εκκρεμές που θα δούμε παρακάτω.

2 Μετατροπή δευτεροβάθμιας εξίσωσης σε σύστημα πρώτου βαθμού (εξίσωση για το απλό εκκρεμές).

Ένα απλό εκκρεμές αποτελείται από σημειακή μάζα στο άκρο μιάς ράβδου μήκους L που στηρίζεται σε κάποιο καρφί (χωρίς τριβή). Αν η βαρύτητα είναι η μόνη δύναμη που ενεργεί τότε η ταλάντωση του εκκρεμούς διαμορφώνεται από την εξίσωση

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin(\theta), \tag{3}$$

όπου θ είναι η γωνιακή θέση της ράβδου, με $\theta = 0$ αν η ράβδος κρέμεται κάτω από το καρφί και $\theta = \pi$ αν η ράβδος ξεκινάει τη ταλάντωση ακριβώς πάνω από το καρφί. Πάρτε $L = 50cm$ και $g = 9.81m/s^2$. Οι αρχικές συνθήκες είναι

$$\theta(0) = \theta_0, \quad \text{και} \quad \frac{d\theta}{dt}(0) = 0. \tag{4}$$

Εάν η αρχική γωνία θ_0 δεν είναι αρκετά μεγάλη, τότε η προσέγγιση $\sin(\theta) = \theta$ μπορεί να χρησιμοποιηθεί και οδηγεί στο γραμμικό μοντέλο του ταλαντωτή

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\theta, \tag{5}$$

το οποίο λύνεται έυκολα και με τις αρχικές συνθήκες που έχουμε η ακριβής λύση είναι

$$\theta(t) = \theta_0 \cos(t\sqrt{g/L}). \tag{6}$$

Η εξίσωση (5) μπορεί να γραφτεί ως σύστημα 1ης τάξης κάνοντας την εξής αλλαγή μεταβλητών: $y_1 = \theta$ και $y_2 = \dfrac{d\theta}{dt}$ οπότε και προκύπτει το σύστημα

$$\begin{cases} \dfrac{dy_1}{dt} = y_2, \\ \dfrac{dy_2}{dt} = -\dfrac{g}{L}y_1, \end{cases} \tag{7}$$

ή αλλιως

$$\frac{dy}{dt} = Ay \text{ με } y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \text{ και } A = \begin{pmatrix} 0 & 1 \\ -\dfrac{g}{L} & 0 \end{pmatrix}. \tag{8}$$

3 Υλοποίηση της μεθόδου Euler για συστήματα

Παίρνοντας αφορμή από το παραπάνω παράδειγμα ας προσπαθήσουμε να υλοποιήσουμε με τη βοήθεια της MatLab τη μέθοδο Euler στο σύστημα (8). Ξεκινάμε φτιάχνοντας ένα αρχείο `myeuler.m` που θα περιέχει το κώδικα

```
1  function [t,y]=myeuler(a,b,ya,N)
2  h=(b-a)/N;
3  t=zeros(1,N+1);
4  y=zeros(2,N+1);
5  t(1)=a;
6  y(:,1)=ya;
7  for i=1:N
8      y(:,i+1)=y(:,i)+h*f(t(i),y(:,i));
9      t(i+1)=t(i)+h;
10 end
```

Ας παρατηρήσουμε εδώ πως, σε αντίθεση με την συνάρτηση `myeuler.m` που υλοποιήσαμε στο προηγούμενο εργαστήριο, το `y` πλέον έχει διάσταση $2 \times N$, διάσταση την οποία πρέπει να έχει και η αρχική τιμή της εξίσωσης, `ya`. Στην γραμμή 6, αναθέτουμε το διάνυσμα αρχικών τιμών, στην πρώτη στήλη του `y`, χρησιμοποιώντας την σύνταξη

```
y(:,1)=ya;
```

η οποία υποδηλώνει πως θα αντικαταστήσουμε ολόκληρη την πρώτη *στήλη* του `y` με το `ya`. Αντίστοιχα, μέσα στην επανάληψη και συγκεκριμένα στην γραμμή 8, δουλεύουμε με ολόκληρες τις στήλες i και $i+1$ του `y`.

Μετά, θα δημιουργήσουμε ένα αρχείο f.m το οποίο θα περιέχει το δεξί μέλος του συστήματος, και θα περιγράφεται από τον εξής κώδικα:

```
1  function [z]=f(t,y)
2  L=0.5;
3  g=9.81;
4  z = zeros(size(y));
5  z(1)=y(2);
6  z(2)=-(g/L)*y(1);
```

Επίσης, για την ακριβή λύση θα φτιάξουμε δύο αρχεία *exact_th.m* και *exact_th_t.m* για την θ και την $\dfrac{d\theta}{dt}$ αντίστοιχα.

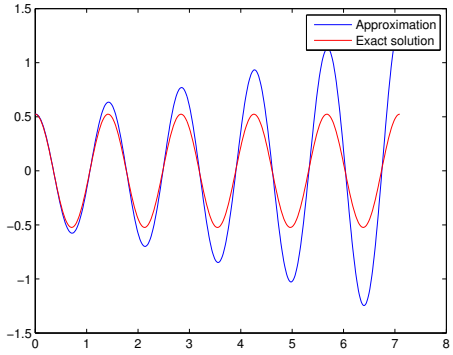
```
1  function [z]=exact_th(t,th0)
2  L=0.5;
3  g=9.81;
4  z=th0*cos(sqrt(g/L)*t);
```

```
1  function [z]=exact_th_t(t,th0)
2  L=0.5;
3  g=9.81;
4  z=-th0*sqrt(g/L)*sin(sqrt(g/L)*t);
```

Για να υπολογίσουμε τις προσεγγίσεις y^n της $y(t)$, εκτελούμε ένα μικρό script με τις παρακάτω εντολές:

```
1  clear all;
2  th0=pi/6;
3  y0=[th0,0];
4  T=1.4185;
5  t0=0.0;
6  N=512;
7  [t,y]=myeuler(t0,5*T,y0,N);
8  plot(t,y(1,:),t,exact_th(t,th0),'r')
9  legend('Approximation','Exact solution')
```

οπότε και προκύπτει το παρακάτω γράφημα



Σχήμα 1: Προσσεγιστική λύση με τη μέθοδο του Euler (μπλέ γραμμή) και ακριβής λύση (κόκκινη γραμμή), για την εξίσωση (8), για $N = 512$.

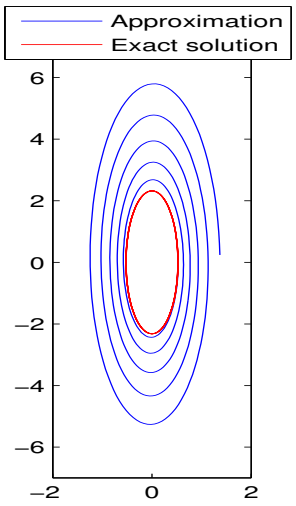
Να παρατηρήσουμε πως παρόλο που υποβαθμίσαμε την δεύτερης τάξης εξίσωση σε ένα σύστημα δύο εξισώσεων πρώτης τάξης, η λύση που ζητάμε είναι αυτό που αντιστοιχεί στην πρώτη στήλη του διανύσματος y ή, αντίστοιχα, το y_1 όπως έχει δηλωθεί στο σύστημα (8).

4 Διάγραμμα φάσης

Το διάγραμμα φάσης μπορεί να φτιαχτεί με τις εντολές

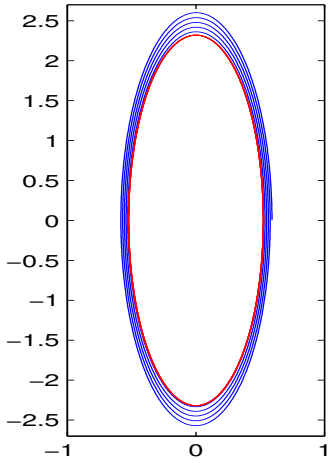
```
1 >>plot(exact_th(t,th0),exact_th_t(t,th0),'r')
2 hold on
3 plot(y(1,:),y(2,:),'b')
```

και βλέπουμε το αποτέλεσμά τους στο Σχήμα 2. Παρατηρούμε πως για την ακριβή λύση έχουμε μία μόνο τροχιά, ενώ το διάγραμμα φάσης για την προσεγγιστική μας λύση είναι μία σπείρα που εκτείνεται προς τα έξω.



Σχήμα 2: Διάγραμμα φάσης της λύσης με τη μέθοδο του Euler (μπλέ γραμμή) και της ακριβής λύση (κοκκινη γραμμή), για την εξίσωση (6), για $N = 512$.

Το επόμενο βήμα μας είναι να χρησιμοποιήσουμε περισότερα σημεία για την προσέγγιση της λύσης μας και συγκεκριμένα χρησιμοποιούμε $N = 4096$ σημεία. Τότε παρατηρούμε πως η σπείρα που περιγράφει το διάγραμμα φάσης της προσέγγισής μας είναι πιο συγκεντρωμένο.



Σχήμα 3: Διάγραμμα φάσης της λύσης με τη μέθοδο του Euler (μπλέ γραμμή) και ακριβής λύση (κόκκινη γραμμή), για το σύστημα (8), για $N = 4096$.

5 Οι συναρτήσεις ode23 και ode45

Η Matlab, και συγκεκριμένα το Ordinary Differential Equations Toolbox περιέχει ένα μεγάλο αριθμό από αλγορίθμους για την αριθμητική επίλυση συνήθων διαφορικών εξισώσεων. Σε αυτές τις σημειώσεις, θα μιλήσουμε για 2 συγκεκριμένες μεθόδους, τις `ode23` και `ode45`. Οι μέθοδοι αυτοί υλοποιούν δύο μεθόδους Runge-Kutta που προσαρμόζουν το βήμα τους ώστε να έχουν ένα επιθυμητό τοπικό και απόλυτο σφάλμα. Η `ode23` εφαρμόζει μεθόδους R-K τάξης 2 και 3, ενώ η `ode45` τάξης 4 και 5. Η συνάρτηση `ode45` χρησιμοποιείται πιο συχνά, ενώ η `ode23` χρησιμοποιείται στην περίπτωση που προσπαθούμε να λύσουμε ένα άκαμπτο σύστημα. Αυτό γίνεται διότι λόγω της υψηλής τάξης των μεθόδων που χρησιμοποιεί η `ode45`, ο χρόνος υπολογισμού της λύσης ενός άκαμπτου συστήματος είναι πολύ μεγάλος.

Εδώ θα δείξουμε ένα παράδειγμα για το πώς εκτελείται η συνάρτηση `ode45`. Ο τρόπος εκτέλεσης της `ode23` είναι ακριβώς ο ίδιος (η μόνη διαφορά είναι η τάξη ακρίβειας των μεθόδων που χρησιμοποιούνται). Η `ode45` λοιπόν καλείται ως

```
>> [t,y] = ode45(odefun,tspan,y0);,
```

όπου `odefun` πρέπει να επιστρέφει ένα διάνυσμα στήλη με τιμές που αντιστοιχούν στην συνάρτηση $f(t, y)$, ενώ `tspan` είναι το διάστημα στο οποίο υπολογίζουμε την προσέγγισή μας και `y0` είναι η αρχική τιμή. Σε περίπτωση που θέλουμε να επιβάλουμε συγκεκριμένες τιμές για το απόλυτο και σχετικό σφάλμα, τότε μπορούμε να εκτελέσουμε την `ode45` ως

```
>> options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4]);
>> [t,y] = ode45(odefun,tspan,y0,options);
```

Με την εντολή `odeset` ορίζουμε κάποιες ιδιότητες τις οποίες θέλουμε να ικανοποιούνται κατά την εκτέλεση της `ode45`. Για μία λίστα με όλες τις παραμέτρους που μπορούμε να αλλάξουμε, μπορείτε να συμβουλευτείτε τη βάση δεδομένων της Matlab για την συγκεκριμένη συνάρτηση, με την εντολή `doc ode45`. Οι προεπιλεγμένες τιμές για το σχετικό και απόλυτο σφάλμα των μεθόδων είναι 10^{-3} και 10^{-6} αντίστοιχα.

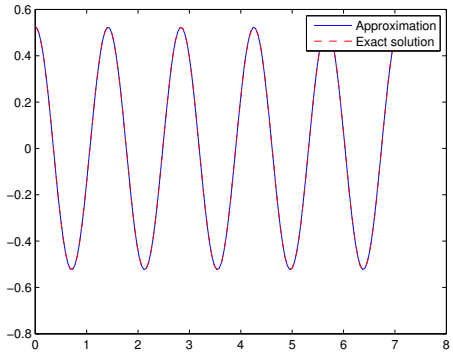
Για να δοκιμάσουμε την συνάρτηση στο πρόβλημα που παρουσιάσαμε προηγουμένως, αρκεί να αλλάξουμε στην γραμμή 7 του script και να βάλουμε στη θέση της την γραμμή

```
[t,y] = ode45(@f,[0,5*T],y0);
```

Επίσης, επειδή η συνάρτηση `ode45` επιστρέφει διανύσματα στήλες ενώ ο κώδικας για την μέθοδο Euler που υλοποιήσαμε πριν επιστρέφει διανύσματα γραμμές, πρέπει να τροποποιήσουμε και την γραμμή που αντιστοιχεί στην εκτύπωση της γραφικής παράστασης, έτσι ώστε πλέον να σχεδιάζει την πρώτη *στήλη* του `y` και όχι την πρώτη *γραμμή*. Αυτό γίνεται απλά γράφοντας

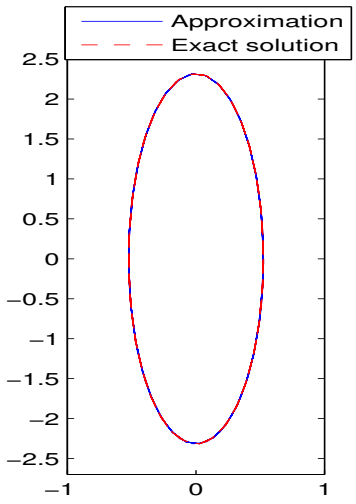
```
plot(t,y(:,1),t,exact_th(t,th0),'r--')
```

Τώρα, αν σχεδιάσουμε την προσεγγιστική και ακριβή λύση, όπως φαίνεται στο Σχήμα 4, βλέπουμε πως οι 2 λύσεις συμπίπτουν ποιοτικά. Να τονίσουμε επίσης πως με την χρήση της ode45, χρειαστήκαμε μόνο 185 βήματα για να φτάσουμε στην λύση, ενώ όπως μπορούμε να δούμε και στο Σχήμα 3, η μέθοδος του Euler με 512 σημεία δεν προσεγγίζει την ακριβή λύση καθόλου καλά.



Σχήμα 4: Προσσεγιστική λύση με τη μέθοδο ode45 (μπλέ γραμμή) και ακριβής λύση (κόκκινη γραμμή), για το σύστημα (8).

Τέλος, αν δούμε το Σχήμα 5, είναι εμφανές πως η προσέγγισή μας είναι αρκετά καλή και όσον αφορά την πρώτη παράγωγο της λύσης μας, αφού τα διαγράμματα φάσης της προσεγγιστικής και ακριβούς λύσης μας συμπίπτουν ποιοτικά.



Σχήμα 5: Διάγραμμα φάσης της λύσης με τη μέθοδο ode45 (μπλέ γραμμή) και ακριβής λύση (κόκκινη γραμμή), για το σύστημα (8).