Λύση Συνήθων Διαφορικών Εξισώσεων με το Matlab

Το MATLAB και συγκεκριμένα το Ordinary Differential Equations toolbox παρέχει ένα μεγάλο αριθμό από αλγορίθμους για την αριθμητική επίλυση συνήθων διαφορικών εξισώσεων. Σε αυτές τις σημειώσεις θα περιγράψουμε μία από αυτές τις μεθόδους, την ode23. Για τις υπόλοιπες μεθόδους, συμβουλευτείται το εγχειρίδιο χρήσης του MATLAB. Ο τρόπος χρήσης τους είναι ανάλογος αλλά μερικές από αυτές έχουν είτε μεγαλύτερη τάξη ακρίβειας είτε/και είναι κατάλληλες για άκαμπτες διαφορικές εξισώσεις.

Μας ενδιαφέρει το ακόλουθο πρόβλημα αρχικών τιμών: έστω $m \geq 1$ και $f : [a, b] \times \mathbb{R}^m \to \mathbb{R}^m$ μια αρκετά ομαλή συνάρτηση και $y_0 \in \mathbb{R}^m$. Ζητείται $y : [a, b] \times \mathbb{R}^m \to \mathbb{R}^m$ τέτοια ώστε

$$\begin{cases} y'(t) = f(t, y(t)), & a \leq t \leq b, \\ y(a) = y_0. \end{cases} \tag{1}$$

Η συνάρτηση ode23 χρησιμοποιεί δύο εμφυτευμένες μεθόδους Runge–Kutta τάξης δύο, αντίστοιχα, τρία, για τη λύση του (1), την εκτίμηση του τοπικού σφάλματος και την επιλογή του βήματος (δείτε τις σημειώσεις του μαθήματος γιά ένα ακόμα τρόπο αυτόματης επιλογής του βήματος). Η μέθοδος έχει τρία στάδια (τα συμβολίζουμε εδώ με s_1, s_2, s_3) και ορίζεται από τις σχέσεις:

$$\begin{aligned} s_1 &= f(t^n, y^n), \\ s_2 &= f(t^n + \frac{h}{2}, y^n + \frac{h}{2} s_1), \\ s_3 &= f(t^n + \frac{3h}{4}, y^n + \frac{3h}{4} s_2), \\ y^{n+1} &= y^n + \frac{h}{9}(2s_1 + 3s_2 + 4s_3). \end{aligned}$$

Είναι εύκολο να δεί κανείς ότι μια καλή εκτίμηση του τοπικού σφάλματος δίνεται από τις σχέσεις

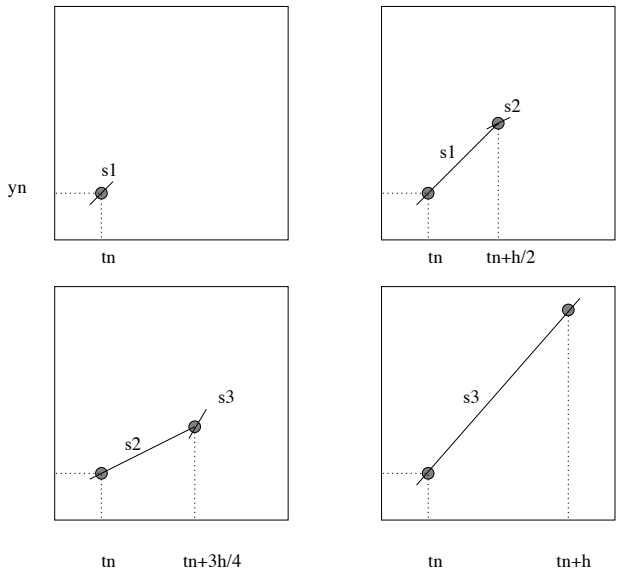
$$\begin{aligned} s_4 &= f(t^{n+1}, y^{n+1}), \\ \delta^n &= \frac{h}{72}(-5s_1 + 6s_2 + 8s_3 - 9s_4). \end{aligned}$$

Ο αλγόριθμος ξεκινάει από το σημείο (t^n, y^n) με μιά αρχική κλίση $s_1 = f(t^n, y^n)$ και μιά αρχική εκτίμηση, h, του βήματος. Το πρώτο στάδιο της μεθόδου είναι η μέθοδος του Euler με βήμα $h/2$. Το δεύτερο στάδιο της μεθόδου είναι πάλι η μέθοδος του Euler αλλά με βήμα $3h/4$. Η τελική εκτίμηση y^{n+1} είναι ένας γραμμικός συνδιασμός των ενδιάμεσων σταδίων. Η ερμηνεία των ενδιάμεσων σταδίων ως κλίσεις, φαίνεται στο Σχήμα 1. Το τοπικό σφάλμα εκτιμάται από την ποσότητα

$$\delta^n = \frac{h}{72}(-5s_1 + 6s_2 + 8s_3 - 9s_4).$$

Στόχος μας είναι να υπολογίσουμε μια προσέγγιση y^{n+1} της λύσης στο χρόνο $t^{n+1} = t^n + h$ έτσι ώστε $|y^{n+1} - y(t^{n+1})| \leq \epsilon$, όπου ϵ είναι μια μικρή σταθερά. Άν $|\delta^n| \leq \epsilon$ τότε δεχόμαστε την εκτίμηση y^{n+1} και προχωράμε στο επόμενο βήμα. Διαφορετικά, απορρίπτουμε την εκτίμηση y^{n+1}, υποδιπλασιάζουμε το βήμα h και επιχειρούμε πάλι τον υπολογισμό της προσέγγισης y^{n+1}. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου γίνει αποδεκτή η προσέγγιση y^{n+1}.

Η συνάρτηση ode23 του Matlab είναι κατάλληλη για τη λύση μη άκαμπτων διαφορικών εξισώσεων της μορφής (1). Στην απλούστερή της μορφή, η συνάρτηση ode23 καλείται ως



Σχήμα 1: Ο αλγόριθμος της συνάρτησης ode23.

```
[T,Y] = ode23(ODEFUN, TSPAN, Y0);
```

Αν `TSPAN = [T0 TFINAL]` τότε η ode23 ολοκληρώνει τη διαφορική εξίσωση του προβλήματος (1) από το `T0` μέχρι το `TFINAL`, με αρχική τιμή `Y0`. Η συνάρτηση `ODEFUN(T,Y)` πρέπει να επιστρέφει ένα διάνυσμα στήλη με τις τιμές που αντιστοιχούν στη συνάρτηση $f(t, y)$. Η συνάρτηση `ODEFUN` μπορεί να δοθεί είτε σαν inline συνάρτηση είτε να οριστεί σε ένα M–file (ακολουθούν παραδείγματα). Η συνάρτηση ode23 επιστρέφει το διάνυσμα στήλη `T` και τον πίνακα `Y`. Η γραμμή i του πίνακα `Y` περιέχει τη λύση στη χρονική στιγμή `T(i)`.

Παράδειγμα 1. Για να λύσουμε το πρόβλημα αρχικών τιμών

$$\begin{cases} y'(t) = 4t - 3y, & 0 \le t \le 1, \\ y(0) = 2, \end{cases}$$

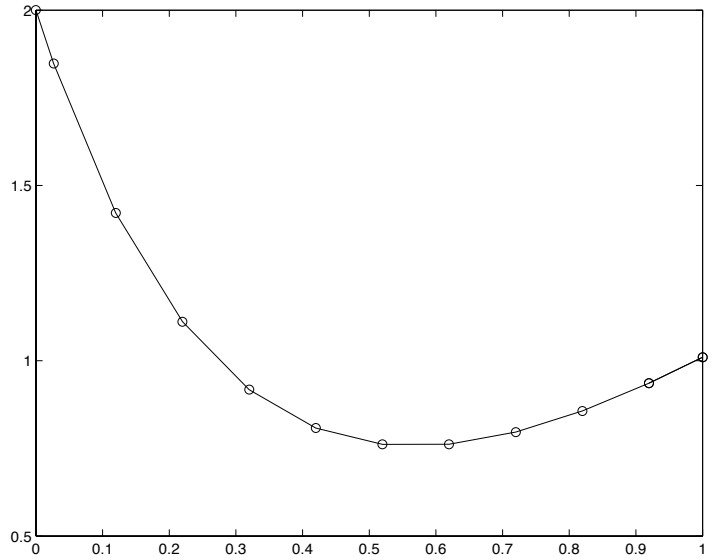
μπορούμε να χρησιμοποιήσουμε τις εντολές

```
>> f=inline('4*t-3*y', 't', 'y');
>> [t,y]=ode23(f, [0 1], 2);
>> plot(t, y);
```

Η πρώτη από τις παραπάνω εντολές ορίζει το δεξί μέλος της διαφορικής εξίσωσης. Επειδή εδώ η συνάρτηση είναι πολύ απλή την γράφουμε σαν inline συνάρτηση. Η τελευταία εντολή σχεδιάζει τη λύση στο διάστημα $[0, 1]$. Αν δεν μας ενδιαφέρουν τα διανύσματα `T` και `Y` που επιστρέφει η ode23 μπορούμε απλά να γράψουμε

```
>> f=inline('4*t-3*y', 't', 'y');
>> ode23(f, [0 1], 2);
```

και το Matlab θα σχεδιάσει τη λύση ενώ λύνει τη διαφορική εξίσωση. Αν όλα πήγαν καλά, θα δείτε τη γραφική παράσταση που φαίνεται στο Σχήμα 2. Εναλλακτικά, θα μπορούσαμε να γράψουμε τη συνάρτηση `f` σε ένα M–file με το όνομα `f.m` που να περιέχει τις εντολές



Σχήμα 2: Η λύση της ode23 για το Παράδειγμα 1.

```
function ydot = f(t,y)
   ydot = 4*t-3*y;
```

και να λύσουμε το πρόβλημα αρχικών τιμών με την εντολή

```
>> ode23(@f, [0 1], 2);
```

Άσκηση 1. Λύστε το πρόβλημα αρχικών τιμών (1) στο διάστημα $[0, 10]$ με $y(0) = 1$ και δεξί μέλος

1. $f(t, y) = 0$,
2. $f(t, y) = t$,
3. $f(t, y) = y$,
4. $f(t, y) = 1/(1 - 3t)$,
5. $f(t, y) = 2y - y^2$.

Συγκρίνετε με την ακριβή λύση.

Παράδειγμα 2. Σε αυτό το παράδειγμα θα λύσουμε το λεγόμενο "πρόβλημα των δύο σωμάτων" (two-body problem). Αυτό περιγράφει την τροχιά ενός σώματος A κάτω από την επίδραση της βαρυτικής έλξης γύρω από ένα κατά πολύ βαρύτερο σώμα B. Αν $(u(t), v(t))$ είναι οι συντεταγμένες του σώματος A στη χρονική στιγμή t τότε

$$u''(t) = -u(t)/r^3(t),$$
$$v''(t) = -v(t)/r^3(t),$$

όπου $r(t) = \sqrt{u^2(t) + v^2(t)}$. Μπορούμε να γράψουμε το παραπάνω σύστημα εξισώσεων σαν ένα σύστημα εξισώσεων πρώτης τάξης αν θέσουμε

$$y(t) = \begin{bmatrix} u(t) \\ v(t) \\ u'(t) \\ v'(t) \end{bmatrix}$$

Η $y(t)$ ικανοποιεί τότε τη διαφορική εξίσωση

$$y'(t) = \begin{bmatrix} u'(t) \\ v'(t) \\ -u(t)/r^3(t) \\ -v(t)/r^3(t) \end{bmatrix} = \begin{bmatrix} y_3(t) \\ y_4(t) \\ y_1(t)/r^3(t) \\ y_2(t)/r^3(t) \end{bmatrix}$$

Γράφουμε στο M–file `twobody.m` τις εντολές

```
function ydot = twobody(t,y)
   r = sqrt(y(1)^2 + y(2)^2);
   ydot = [y(3); y(4); -y(1)/r^3; -y(2)/r^3];
```

και λύνουμε με την ode23 ως συνήθως:

```
>> [t,y] = ode23(@twobody, [0 2*pi], [1; 0; 0; 1]);
>> plot(y(:,1), y(:,2), '-', 0, 0, 'ro');
>> axis([-1.2 1.2 -1.2 1.2]);
```

Η εντολή `plot` όπως είναι γραμμένη παραπάνω σχεδιάζει την τροχιά του σώματος A για $t \in [0, 2\pi]$ και αναπαριστά το σώμα B με ένα μικρό κύκλο στο σημείο $(0, 0)$. Παρατηρήστε ότι το διάνυσμα αρχικών τιμών λέει ότι το σώμα A βρίσκεται γιά $t = 0$ στο σημείο $(1, 0)$ και το διάνυσμα της ταχύτητάς του είναι $(0, 1)$. Αλλάξτε το διάνυσμα αρχικών τιμών και παρατηρήστε πως αλλάζει η τροχιά του σώματος A γύρω από το σώμα B.

Η συνάρτηση ode23 επιτρέπει στον χρήστη να αλλάξει ορισμένες παραμέτρους του αλγορίθμου, όπως τη σταθερά ϵ που φράσει το μέγιστο τοπικό σφάλμα, μέσω ενός τέταρτου προαιρετικού ορίσματος `OPTIONS`:

```
OPTIONS = ODESET('NAME1', VALUE1, 'NAME2', VALUE2, ...);
[T,Y] = ode23(ODEFUN, TSPAN, Y0, OPTIONS);
```

Εδώ, η συνάρτηση `ODESET` του Matlab αναθέτει στις παραμέτρους `NAME1`, `NAME2`, ..., τις τιμές `VALUE1`, `VALUE2`, ..., αντίστοιχα. Μερικές χρήσιμες παράμετροι είναι οι ακόλουθες (για τον πλήρη κατάλογο των παραμέτρων συμβουλευτείται το εγχειρίδιο χρήσης του Matlab):

- `RelTol`. Σχετικό μέγιστο σφάλμα. Θετική σταθερά με τιμή `1.0e-3`. Η ode23 προσπαθεί να διαλέξει το βήμα της μεθόδου έτσι ώστε το σφάλμα `e(i)` για τη συνιστώσα $y_i(t)$ της λύσης να ικανοποιεί

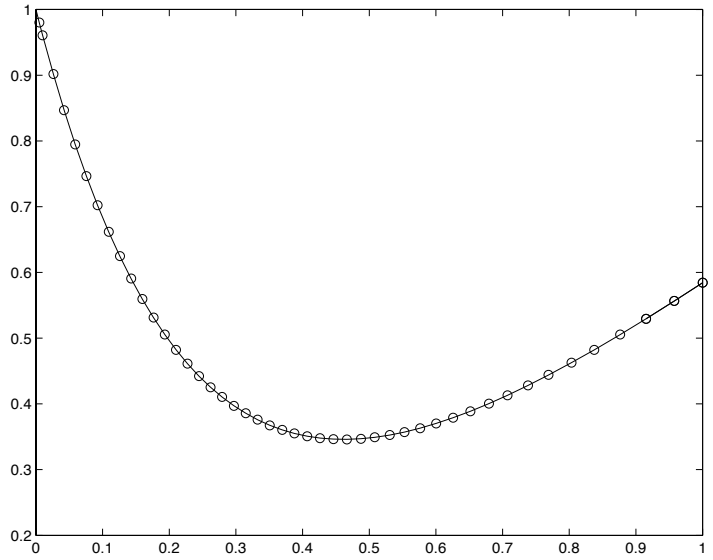
  ```
  e(i) <= max(RelTol*abs(y(i)), AbsTol(i))
  ```

- `AbsTol`. Απόλυτο μέγιστο σφάλμα. Θετική σταθερά ή διάνυσμα με μήκος m και τιμή `1.0e-6`.

- `Refine` Θετική ακέραια σταθερά με τιμή ένα. Αυτή η σταθερά αυξάνει την πυκνότητα των σημείων που χρησιμοποιούνται στη γραφική παράσταση της λύσης κατά το δοσμένο συντελεστή.
- `InitialStep`. Προτεινόμενο αρχικό βήμα.
- `MaxStep`. Μέγιστο επιτρεπτό βήμα. Αν δεν τεθεί από τον χρήστη τότε αυτό υπολογίζεται σαν το ένα δέκατο του μήκους του διαστήματος ολοκλήρωσης.

Παράδειγμα 3. Λύνουμε το πρόβλημα αρχικών τιμών $y'(t) = 3t - 4y$, $0 \leq t \leq 1$ με αρχική τιμή $y(0) = 1$:

```
>> f=inline('3*t-4*y', 't', 'y');
>> options = odeset('RelTol', 1.0e-4, 'Refine', 2, 'InitialStep', 1.0e-2);
>> ode23(f, [0 1], 1, options);
```



Σχήμα 3: Η γραφική παράσταση της λύσης του Παραδείγματος 3.