Αριθμητικη Επιλυση Μερικων Διαφορικων Εξισωσεων

2ο Εργαστηριο

23 Οκτωβρίου 2015

1 Μέθοδοι Πεπερασμένων Διαφορών για παραβολικά προβλήματα

Σκοπός αυτού του εργαστηρίου είναι η υλοποίηση μίας αριθμητικής μεθόδου για την επίλυση μίας παραβολικής εξίσωσης και συγκεκριμένα θέλουμε να προσεγγίσουμε τη λύση της εξίσωσης της θερμότητας. Θα ξεκινήσουμε θεωρώντας το πρόβλημα αρχικών/συνοριακών τιμών για την εξίσωση της θερμότητας, με ομογενείς συνοριακές συνθήκες τύπου Dirichlet

$$\begin{cases} u_t(t,x) = u_{xx}(t,x), & x \in [a,b], \quad t \in [t_0, T_f], \\ u(0,x) = u_0(x), & x \in [a,b], \\ u(t,a) = u(t,b) = 0, & t \in [t_0, T_f], \end{cases} \tag{1}$$

όπου $u_0(x) \in C([a,b])$.

Θέλουμε να χρησιμοποιήσουμε μία μέθοδο πεπερασμένων διαφορών για το πρόβλημα (1), οπότε, χρησιμοποιώντας τις τεχνικές που παρουσιάσαμε στο προηγούμενο εργαστήριο, μπορούμε να προσεγγίσουμε τις παραγώγους από πηλίκα διαφορών. Γι΄ αυτό τον σκοπό, θα διαμερίσουμε το $[a,b]$ σε N_x διαστήματα με μήκος $h = (b-a)/N_x$, όπου τα σημεία $x_i,\ i = 1, \ldots, N_x + 1$ δίνονται από τον τύπο $x_i = a + (i-1)h$, οπότε έχουμε

$$a = x_1 < x_2 < \ldots < x_{N_x} < x_{N_x+1} = b,$$

και αντίστοιχα δημιουργούμε μία διαμέριση του $[t_0, T_f]$ σε N_t διαστήματα μήκους $\tau = (T_f - t_0)/N_t$ και τα σημεία είναι $t_n = t_0 + (n-1)\tau,\ n = 1, \ldots, N_t + 1$. Έχουμε δηλαδή

$$t_0 = t_1 < t_2 < \ldots < t_{N_t} < t_{N_t+1} = T_f.$$

Έχοντας τις διαμερίσεις σε χώρο και χρόνο, μπορούμε τώρα να προσεγγίσουμε τις αντίστοιχες παραγώγους. Στο συγκεκριμένο εργαστήριο θα υλοποιήσουμε την άμεση μέθοδο του Euler, η

οποία δημιουργείται παίρνοντας μία προς τα εμπρός διαφορά για την χρονική παράγωγο u_t και κεντρική διαφορά για την χωρική δεύτερη παράγωγο u_{xx}. Έχουμε δηλαδή για ένα χρόνο t^n και ένα σημείο x_i:

$$u_t(t^n, x_i) \approx \frac{u(t^{n+1}, x_i) - u(t^n, x_i)}{\tau}$$
$$u_{xx}(t^n, x_i) \approx \frac{u(t^n, x_{i+1}) - 2u(t^n, x_i) + u(t^n, x_{i-1})}{h^2}.$$

Περνώντας τις εκφράσεις για τις προσεγγίσεις των παραγώγων στην εξίσωση (1), έχουμε

$$\frac{u(t^{n+1}, x_i) - u(t^n, x_i)}{\tau} = \frac{u(t^n, x_{i+1}) - 2u(t^n, x_i) + u(t^n, x_{i-1})}{h^2}$$

Σκοπός μας είναι να βρούμε προσεγγίσεις U_i^n των $u(t^n, x_i)$, οπότε δημιουργούμε το σύστημα εξισώσεων

$$\frac{U_i^{n+1} - U_i^n}{\tau} = \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{h^2}, \quad i = 2, \dots N_x,\ n = 1, \dots N_t + 1,$$

και εισάγοντας $\mu = \tau/h^2$ τη σταθερά Courant, καταλήγουμε στην σχέση για την άμεση μέθοδο του Euler για την εξίσωση (1)

$$U_i^{n+1} = \mu U_{i+1}^n + (1 - 2\mu)U_i^n + \mu U_{i-1}^n, \quad i = 2, \dots N_x,\ n = 1, \dots N_t + 1. \tag{2}$$

2 Υλοποίηση στη Matlab

Το επόμενο βήμα μας είναι να λύσουμε αριθμητικά την εξίσωση (1) για $[a, b] = [0, 1]$ και $[t_0, T_f] = [0, 0.25]$. Αρχίζουμε λοιπόν δημιουργώντας το αρχείο μας και ορίζοντας το χωρίο στο οποίο θα λύσουμε την εξίσωση.

```
a  = 0;
b  = 1;
t0 = 0;
T  = .25;
```

Θα δημιουργήσουμε επίσης τα μήκη των διαστημάτων και την σταθερά Courant έτσι ώστε να μη μας δημιουργήσει κάποιο πρόβλημα. Στο συγκεκριμένο παράδειγμα, ορίζουμε το N_x και θέτουμε τη σταθερά $\mu = 0.5$. Εφόσον ξέρουμε ότι $\mu = \tau/h^2$, είναι εύκολο να υπολογίσουμε από εκει το τ και έπειτα το N_t. Επειδή όμως το N_t που θα προκύψει μπορεί να μήν ακέραιο, παίρνουμε το πάνω ακέραιο μέρος του, έτσι ώστε το τ' που θα προκύψει από το $\lceil N_t \rceil$ να μας δημιουργήσει ένα $\mu' \leq \mu$.

```
Nx = 52;

x = linspace(a,b,Nx+1);
h = (b-a)/Nx;

mu = 0.5

tau = h*h*mu;
Nt = ceil((T-t0)/tau);

tau = (T-t0)/Nt;

mu = tau/(h*h)
```

Για να αποθηκεύσουμε τις προσεγγίσεις U_i^n δημιουργούμε έναν $(N_x+1)\times(N_t+1)$ πίνακα U, όπου στην $n-$οστή στήλη περιέχει τις προσεγγίσεις της $u(t^n,\cdot)$.

```
U = zeros(Nx+1,Nt+1);
```

Η γενική μας τακτική είναι να γεμίσουμε τον πίνακα με 0, κάτι που μας εξυπηρετεί τώρα, εφόσον οι συνοριακές μας συνθήκες είναι ομογενείς τύπου Dirichlet.
Στη συγκεκριμένη περίπτωση, θα θεωρήσουμε ώς αρχική συνθήκη τη συνάρτηση

$$u_0(x) := \begin{cases} 2x, & x\in[0,1/2] \\ 2-2x, & x\in(1/2,1]. \end{cases}$$

Για να θέσουμε την αρχική μας συνθήκη, γεμίζουμε την πρώτη στήλη του πίνακα U ως εξής: Αναθέτουμε στις μισές πρώτες γραμμές τις τιμές που αντιστοιχούν στον πρώτο κλάδο και μετά γεμίζουμε τις υπόλοιπες με τον δεύτερο κλάδο.

```
U(1:floor(Nx/2),1) = 2*x(1:floor(Nx/2));
U(floor(Nx/2)+1:Nx+1,1) = 2 - 2*x(floor(Nx/2)+1:Nx+1);
```

Πλέον, έχουμε φτάσει στο σημείο που πρέπει να λύσουμε το σύστημα που περιγράφεται στην (2). Αυτό γίνεται με 2 τρόπους: Μπορούμε να υπολογίζουμε τη λύση για κάθε χρονικό βήμα κατά σημείο με τη χρήση 2 `for`, δηλαδή να υπολογίζουμε το

```
for n = 1 : Nt - 1
        for i = 2 : Nx
                U(i,n+1) = mu*U(i+1,n) + (1-2*mu)*U(i,n) + mu*U(i-1,n);
        end
end
```

ή εναλλακτικά να υπολογίζουμε την προσέγγιση σε όλα τα σημεία ταυτόχρονα, χρησιμοποιώντας διανύσματα, δηλαδή να υπολογίζουμε

$$U_{\cdot}^{n+1} = AU_{\cdot}^{n}, \quad n = 1, \ldots, N_t,$$

όπου

$$A = \begin{pmatrix} 1-2\mu & \mu & 0 & \cdots & 0 \\ \mu & 1-2\mu & \mu & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \mu & 1-2\mu & \mu \\ 0 & \cdots & 0 & \mu & 1-2\mu \end{pmatrix}, \quad U_{\cdot}^{n} = \begin{pmatrix} U_2^n \\ U_3^n \\ \vdots \\ U_{N_x-1}^n \\ U_{N_x}^n \end{pmatrix},$$

κάτι το οποίο θα εφαρμόσουμε στη Matlab ως εξής:

```
dd = (1-2*mu)*ones(Nx-1,1);
du = mu*ones(Nx-2,1);
dl = mu*ones(Nx-2,1);

A = diag(dd) + diag(du,1) + diag(dl,-1);

for n = 1 : Nt - 1
    U(2:Nx,n+1) = A*U(2:Nx,n);
end
```

Σε αυτό το σημείο να ξεκαθαρίσουμε πως και οι δύο τεχνικές θα μας δώσουνε το ίδιο αποτέλεσμα, με τη μόνη διαφορά είναι ότι η δεύτερη τεχνική θα υπολογίσει τη λύση πιο εύκολα, μιας και θα εκμετελλευτεί την δυνατότητα της Matlab να κάνει πράξεις με διανύσματα πολύ γρήγορα.