

Sysi-Aho Marko
48500 P Tfy III
10.4.2001

S-114.240 Seminar on Computational Engineering

The Finite Difference Method in Partial Differential Equations

Introduction

The ultimate goal of discrete methods is the reduction of continuous systems to equivalent discrete systems which are suitable for computer solution. The basic approximation involves the replacement of a continuous domain D by a pattern, network or mesh of discrete points within D . The only regular polygons which can completely fill the plane are rectangles, triangles and hexagons. These are therefore the most used mesh types in discretization.

The purpose of this paper is to give concrete and rather simple ways to solve some well-known partial differential equations with difference method. This policy differs slightly from that of Gersfelds [1] which instead tend to provide a minimal introduction to different categories of partial differential equations and their numerical solution. To perceive the general view, it may be helpful to make oneself familiar with the connection between classes of hyperbolic, parabolic and elliptic equations. This theme is illustrated clearly in Ames [2]. Being familiar with the theory of functional analysis may give a superior view to the thematic at hand. An excellent introduction to functional analysis is provided by Brezis [3].

Some very important issues like errors and convergence of a numerical solution or its existence are not considered seriously. Ames [2] and Mitchell [4] includes a mild survey to these issues and furthermore they provide several references for further studying.

1. Potential equation and its discretization

Let D be a bounded area and B its boundary. Let us consider a problem

$$\begin{aligned} -\Delta u + au &= f & x \in D \subset R \\ u &= g & x \in B \end{aligned}$$

where $a > 0$ is a constant and f, g are continuous functions.

The easiest case occurs in one dimension. Let us consider the following differential equation and how it can be inverted to a simple difference equation.

$$\begin{aligned} -u'' + au &= f & x \in (0,1) \\ u(0) = u(1) &= 0 \end{aligned} \quad (1)$$

An intuitively apparent way to approximate differentials with differences is as follows

$$\frac{u(x+h) - u(x)}{h} \approx u'(x+h/2), \quad \frac{u(x-h) - u(x)}{h} \approx u'(x-h/2)$$

so that

$$u''(x) \approx \frac{u'(x+h/2) - u'(x-h/2)}{h} \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \quad (1D)$$

Let us choose some integer n and set $h = 1/(n+1)$. Let u be the solution of the problem (1) and let u_i denote the approximation of $u(ih)$ and $f_i = f(ih)$. With the above difference equation, we get

$$\begin{aligned}
(-u_0 + (2 + ah^2)u_1 - u_2)/h^2 &= f_1 \\
(-u_1 + (2 + ah^2)u_2 - u_3)/h^2 &= f_2 \\
&\vdots \\
(-u_{n-2} + (2 + ah^2)u_{n-1} - u_n)/h^2 &= f_{n-1} \\
(-u_{n-1} + (2 + ah^2)u_n - u_{n+1})/h^2 &= f_n
\end{aligned}$$

There are n equations and $n+2$ unknowns, but boundary conditions will give $u_0 = u_{n+1} = 0$ and thus there remains only n unknowns. The above group of equations is more conveniently represented in a matrix form.

$$A_h u_h = f_h \quad (2)$$

in which

$$A_h = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix} + aI = -\Delta_h + aI$$

$$u_h = (u_1, \dots, u_n)^T \quad \text{and} \quad f_h = (f_1, \dots, f_n)^T$$

An important task that attaches oneself to every solution seeking process is the proof of the existence of a solution. There is no such a great emphasis on this interesting aspect in this paper but for the above equation (2) it is included.

The matrix A_h is real and symmetric and therefore all its eigenvalues are real.

Furthermore, by Gershgorin's lemma [5], we may deduce that all eigenvalues lie in the $2/h^2 + a$ centered disc with radius $2/h^2$ ie in the interval $[a, a + 4/h^2]$. Thus it is easily seen that the problem has an unique solution at least in case $a > 0$. To prove the existence when $a=0$, it is sufficient to prove that the equation has only a zero solution when $f_h = 0$. Difference equations will give a relation $u_i = (u_{i-1} + u_{i+1})/2$.

Thus u_i is an average of the left and right side neighboring values and consequently one of the following three conditions is valid: I) $u_{i-1} < u_i < u_{i+1}$ II) $u_{i-1} = u_i = u_{i+1}$ or III) $u_{i-1} > u_i > u_{i+1}$, and then by induction 1) $u_0 < u_1 < \dots < u_n$, 2) $u_0 = u_1 = \dots = u_n$ or 3) $u_0 > u_1 > \dots > u_n$

But $u_0 = u_{n+1}$ from which the existence follows.

Example: An attempt to solve the problem (1) numerically with Matlab. Let us make choices $a = 10$ and $f(x) = 1000x^7 - x$. The code is represented in appendix at the end of this paper and it carries a name `semex1.m`. Numerical solutions are plotted in the figure 1. The uppermost curve represents the exact solution. It should be noted that halving the step size reduces the error term approximately to one fourth.

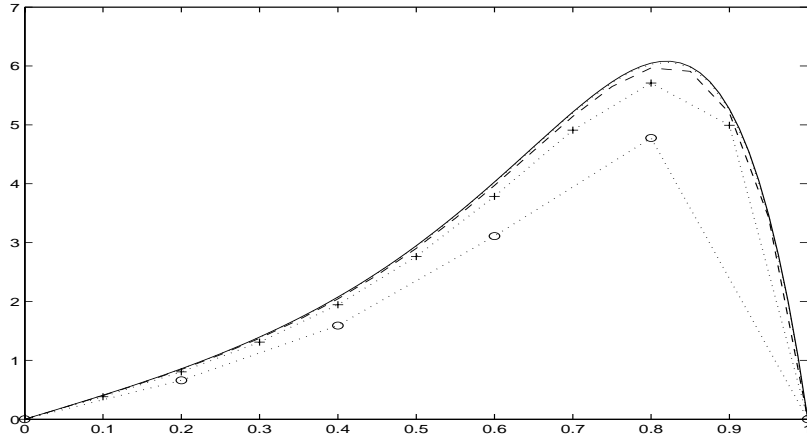


Figure1. Numerical solutions of a problem (1).

Let us next consider a two dimensional problem:

$$\begin{aligned} -\Delta u + au &= f & x \in D = (0,1) \times (0,1) \\ u &= g & x \in B = \partial D \end{aligned} \quad (3A)$$

Let us choose integers n_1, n_2 and set $h_1 = 1/(n_1 + 1), h_2 = 1/(n_2 + 1)$. We denote the approximation of a real solution $u(ih_1, jh_2)$ of the problem (3A) by $u_{i,j}$ and similarly $f_{i,j} = f(ih_1, jh_2)$. Employing the difference approximation (1D) both in x_1 and x_2 directions, we will get a discrete problem:

$$\begin{aligned} -\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_1^2} - \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_2^2} + au_{i,j} &= f_{i,j} & 1 \leq i \leq n_1, 1 \leq j \leq n_2 \\ u_{i,j} &= g_{i,j} & i = 0 \text{ or } j = 0 \text{ or } i = n_1 + 1 \text{ or } j = n_2 + 1 \end{aligned} \quad (3D)$$

Now there are $n_1 n_2$ unknowns. Let us form a vector

$$u_h = (u_{1,1}, u_{2,1}, \dots, u_{n_1,1}, u_{1,2}, \dots, u_{n_1, n_2}) \in R^{n_1 n_2}$$

which can also be written as a matrix

$$U_h = \begin{pmatrix} u_{1,1} & \cdots & u_{1,n_2} \\ \vdots & \ddots & \vdots \\ u_{n_1,1} & \cdots & u_{n_1, n_2} \end{pmatrix} \in R^{n_1 \times n_2}$$

Furthermore let us form a matrix $F_h = [f_{i,j}]$ and matrixes $\Delta_{h_1}, \Delta_{h_2}$ just like previously in a one-dimensional case. If $g=0$, equation (3D) is equivalent to a matrix equation

$$-\Delta_{h_1} U_h - U_h \Delta_{h_2} + a U_h = F_h$$

More generally (3D) can be written as $A_h u_h = b_h$, where

$$A_h = -I_{n_2} \otimes \Delta_{h_1} - \Delta_{h_2} \otimes I_{n_1} + a I_{n_1 n_2}$$

\otimes denotes the Kronecker's product:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n_2}B \\ a_{21}B & a_{22}B & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_1 1}B & a_{n_1 2}B & \cdots & a_{n_1 n_2}B \end{pmatrix} \in R^{n_1 m_1 \times n_2 m_2}, \text{ when } A \in R^{n_1 \times n_2}, B \in R^{m_1 \times m_2}$$

In case $h_1 = h_2$ we get a problem

$$A_h u_h = b_h \quad (4)$$

where

$$A_h = \begin{pmatrix} B & -I & 0 & \cdots & 0 \\ -I & B & -I & \cdots & 0 \\ 0 & -I & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -I \\ 0 & 0 & \cdots & -I & B \end{pmatrix} + aI \quad B = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & \cdots & 0 \\ 0 & -1 & 4 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -1 \\ 0 & 0 & \cdots & -1 & 4 \end{pmatrix}$$

and b depends both on f and g . Again the matrix A is real and symmetric and thus its eigenvalues lie in R . When $a > 0$, we can deduce the existence of a solution similarly as in the one-dimensional case. In the case $a = 0$, a discrete maximum principle can be used to prove the existence of an unique solution.

Example: Let us consider the numerical solution of the problem (3A) in case $a = 1$ and

$$f(x) = \begin{cases} 1, & |x - p| \leq 0.1 \\ 0 & \text{otherwise} \end{cases}$$

$$p = \frac{1}{10}(2, 6) \text{ and } n_1 = n_2 = 28.$$

Matlab implementation `semex2.m` can be found from the appendix. Figure 2 shows the numerical solution.

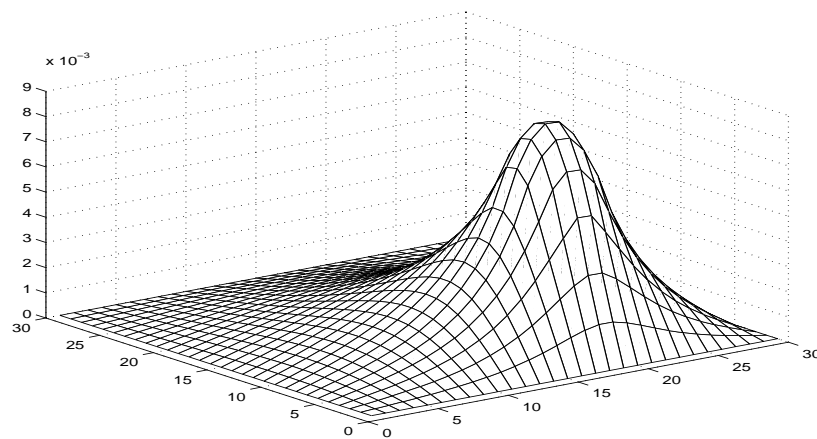


Figure 2. Numerical solution of a problem 3A.

Because matrixes under consideration are sparse and the matrix A is symmetric and positively definite, the linear group of equations (4) can be conveniently solved by the conjugate gradient method (uses matrix and vector products and do not change the coefficient matrix). It is not very practical to try to solve the linear group with a high precision because we have already made approximations by differencing the true differential equation. A more sensible goal is to seek solutions that fall below discretization errors.

The conjugate gradient method:

1. choose x_0 and set $p_0 = r_0 = b - Ax_0$

2. Do while $\|r_k\|$ small enough

$$\alpha_k = \|r_k\|^2 / \langle p_k, Ap_k \rangle$$

$$x_{k+1} = x_k + \alpha_k p_k$$

$$r_{k+1} = r_k - \alpha_k Ap_k$$

$$\beta_k = \|r_{k+1}\|^2 / \|r_k\|^2$$

$$p_{k+1} = r_{k+1} + \beta_k p_k$$

Residuals r_i are orthogonal and thus the exact solution of the equation will be received at least in the n :th iteration when $A \in R^{n \times n}$. In practice, however, n may be such a large number that we would obtain a good approximation earlier.

Let $A \in R^{n \times n}$ be a symmetric and positively definite with all its eigenvalues lying in the interval $[\alpha, \beta]$. Let us solve the problem $Ax = b$ by the conjugate gradient method and denote $e_k = x - x_k$. Then

$$(\langle e_k, Ae_k \rangle)^{1/2} \leq 2 \left(\frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\beta} + \sqrt{\alpha}} \right)^k (\langle e_0, Ae_0 \rangle)^{1/2}$$

Because A is positively definite $\langle e_k, Ae_k \rangle > 0$ when $e_k \neq 0$ and so we are able to define a norm $\|e\|_A = (\langle e, Ae \rangle)^{1/2}$.

Solving partial differential equations with the difference method involves some problems. First of all it is difficult to take curved boundary conditions into consideration. This problem may be avoided by using the finite element method. Another problem concerning both the difference and the element method is that the disturbance propensity of a matrix grows as $h \rightarrow 0$. A common way to approximate the disturbance propensity of the matrix is to use a 1-norm, but when matrixes are positively definite it is justifiable to estimate the behavior with a quantity

$$\kappa_2(A) = \frac{\max \lambda_i}{\min \lambda_i} \quad \lambda_i : s \text{ are eigenvalues of } A$$

By using the Gershgorin's theorem [5] we may deduce that

$$\kappa_2(A) \leq \frac{8 + ah^2}{ah^2} \propto \frac{1}{h^2}$$

When $a=0$, it is still true that $\kappa_2(A) \propto 1/h^2$ and thus the situation will worsen rapidly as the step size h becomes smaller and smaller.

When iterative methods are used, the growth of the disturbance propensity reflects itself as a slow convergence. If we assume $\kappa_2(A) \approx \beta/\alpha$ we will get

$$\|e_k\|_A \leq 2 \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^k \|e_0\|_A$$

Thus the term e_k converges slowly towards zero when the $\kappa_2(A)$ is large.

2. Heat equation and its discretization

Let us consider a one dimensional heat equation

$$\begin{cases} u_t = au_{xx} + f(x,t) & 0 < x < 1 \\ u(0,t) = u(1,t) = 0 & 0 < t < T \\ u(x,0) = g(x) \end{cases} \quad (5)$$

Discretization with respect to x can be done as in the case of the potential equation ie. Choose n and set $h=1/(n+1)$. Let $u_i(t)$ be an approximation of $u(ih,t)$ and let us denote $u_h = (u_1(t), \dots, u_n(t))$ $f_h = (f(h,t), \dots, f(nh,t))$ and $g_h = (g(h,t), \dots, g(nh,t))$. Now we get

$$\begin{cases} u_h'(t) = a\Delta_h u_h(t) + f_h(t) & t \in [0, T] \\ u_h(0) = g_h \end{cases} \quad (6)$$

and

$$\Delta_h = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & 1 & -2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & \dots & 1 & -2 \end{pmatrix}$$

Let us choose a time step d and denote the approximation of $u_h(kd)$ by u_h^k . By using the Euler's method we get a totally discrete problem

$$\begin{cases} u_h^{k+1} = u_h^k + d(a\Delta_h u_h^k + f_h^k) \\ u_h^0 = g_h \end{cases}$$

If $f=0$ the true solution of the heat equation will approach zero but what happens to the numerical solution? When $a=0$, we obtain (by induction)

$$u_h^k = (I + \Delta_h)^k g_h$$

It is true that $B^k \rightarrow 0 \Leftrightarrow \rho(B) < 1$.

In our case $B = I + d\Delta_h$ and all the eigenvalues of Δ_h are lying in the interval $[-4/h^2, 0)$ and thus those of B's in the interval $[1 - 4d/h^2, 1)$. So we get a stability condition $d < h^2/2$; therefore the d must be very small when the h is small. On the other hand parabolic problems tend to become smoother and smoother as time passes on and thus it is reasonable to use implicit methods.

The implicit Euler method (compare with the Euler's method above) gives a discrete problem

$$\begin{cases} u_h^{k+1} = u_h^k + d(a\Delta_h u_h^{k+1} + f_h^{k+1}) \\ u_h^0 = g_h \end{cases}$$

Now there is a linear equation

$$(I - da\Delta_h)u_h^{k+1} = u_h^k + df_h^{k+1} \text{ to be solved in each iteration.}$$

Usually the step size can be chosen to be large enough to get the problem done with less work than by using explicit methods.

For a higher dimensional problem the discretization procedure is quite similar. For example, let us consider the following situation

$$\begin{cases} u_t = a\Delta u + f(x, t) & 0 < x_1, x_2 < 1 \\ u(0, x_2, t) = u(1, x_2, t) = u(x_1, 0, t) = 0 & u_{x_2}(x_1, 1, t) = 0 \\ u(x, 0) = g(x) \end{cases} \quad (7)$$

The Laplace operator can be discretized in a similar way as in the case of the potential equation and the time discretization can be done as previously.

Let Δ_{h_1} be a discrete $\frac{\partial^2}{\partial x_1^2}$ operator with the boundary condition

$$u(0, x_2, t) = u(1, x_2, t) = 0 \text{ and similarly } \Delta_{h_2} \text{ with the boundary condition}$$

$$u(x_1, 0, t) = u_{x_2}(x_1, 1, t) = 0$$

that is

$$\Delta_{h_1} = \frac{1}{h_1^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & 1 & -2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & \dots & 1 & -2 \end{pmatrix} \quad \Delta_{h_2} = \frac{1}{h_2^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & 1 & -2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & \dots & 1 & -2 \end{pmatrix}$$

We get the Laplace operator through a Kronecker product:

$$\Delta_h = I \otimes \Delta_{h_1} + \Delta_{h_2} \otimes I$$

If the time is discretized by the implicit midpoint rule we will get a totally discrete problem

$$\begin{cases} (I - \frac{d}{2} a \Delta_h) u_h^{k+1} = (I + \frac{d}{2} a \Delta_h) u_h^k + d f_h^{k+1/2} \\ u_h^0 = g_h \end{cases},$$

$$u_h^k = (u^{k_{11}}, u^{k_{21}}, \dots, u^{k_{12}}, \dots, u^{k_{n_1 n_2}})$$

and similarly for $f_h^{k+1/2}$ and g_h

Example. The numerical solution of the problem (7). Let $a=1, g=0,$

$$f(x) = \begin{cases} 1, & |x - p| \leq 0.1 \\ 0 & \text{otherwise} \end{cases}, \quad p = \frac{1}{10}(2, 6) \text{ and initial condition } u(x, 0) = 0. \text{ The}$$

matlab version *semex3.m* can be found from the appendix. The time discretization was implemented by the implicit midpoint rule. It is interesting to observe how the solution moves towards the stationary solution ie towards the solution of a corresponding potential equation $\Delta u + f(x, t) = 0$. Figure 3 shows the situation at time $t=2.0$.

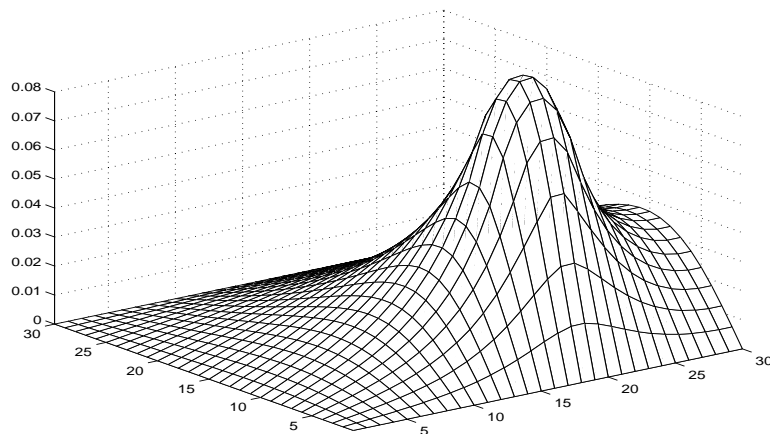


Figure3. The solution at time $t=2.0$.

3. Wave equation and its discretization

Let us consider a one dimensional problem

$$\begin{cases} u_{tt} = c^2 u_{xx} & 0 < x < 1 \\ u(0, t) = u(1, t) = 0 \\ u(x, 0) = f(x), u_t(x, 0) = g(x) \end{cases}$$

As in the case of the heat equation let us first discretize with respect to x and thus we get

$$\begin{cases} u_h'' = c^2 \Delta_h u_h \\ u_h(0) = f_h, u_h' = g_h \end{cases}$$

The group above could be reduced to first order equations but because the equation does not involve first time derivatives, it is profitable to write a straight approximation for the u_h'' . Let us choose time step d and let the approximation of $u_h(kd)$ be u_h^k . By using the same difference equation for the time derivative as for the space derivative we get a discrete problem

$$\frac{u_h^{k+1} - 2u_h^k + u_h^{k-1}}{d^2} = c^2 \Delta_h u_h^k, \text{ with} \quad (8)$$

$$u_h^0 = f_h, \quad u_h^1 = (I + \frac{d^2}{2} \Delta_h) f_h + dg_h$$

Note that both u_h^0 and u_h^1 are needed to start the process. The value u_h^1 is obtained by using the Taylor-series:

$$u_h^1(d) = u_h(0) + u_h'(0)d + \frac{u_h''(0)d^2}{2} + O(d^3) = f_h + g_h d + \frac{d^2}{2} \Delta_h f_h + O(d^3)$$

Problems in higher dimensions can be discretized with similar methods. Take, for example

$$\begin{cases} u_{tt} = c^2 \Delta u & 0 < x_1, x_2 < 1 \\ u(0, x_2, t) = u(1, x_2, t) = u(x_1, 0, t) = 0 & u_{x_2}(x_1, 1, t) = 0 \\ u(x, 0) = f(x), u_t(x, 0) = g(x) \end{cases} \quad (9)$$

The Laplace operator can be discretized just like in the case of the heat equation and time discretization can be done as above. The resulting equation is exactly the same as (8).

Example. The numerical solution of the problem (9). Let us set $c=1$,

$$f(x) = \frac{1}{2} + \frac{1}{2} \cos(5\pi|x-p|), \quad |x-p| \leq 0,2 \text{ and zero otherwise.}$$

$P=(0.6 \ 0.5)$ and $g=0$. Again, the matlab implementation `semex4.m` can be found from the appendix.

Dispersion

Let us consider a problem

$$u_{tt} = c^2 \Delta u$$

where c is the velocity of a signal. Let us seek plane wave solutions ie solutions of type $u(x, t) = e^{i(w'x - Dt)}$ where w is a wave vector and D angular frequency. Inserting this to the above equation will give a dispersion relation $D^2 = c^2 |w|^2$. The phase velocity and the group velocity are defined as follows

$$v_\phi = D / |w| \quad \text{and} \quad v_g = |\nabla D| \quad \text{respectively.}$$

For plane waves both of these velocities are equal to c . The result means that all frequencies propagate with the same velocity to all directions. If either the group or the phase velocity depends on the norm of a wave vector, the equation producing the

dispersion relation is called dispersive. Equations are anisotropic if one or the other of velocities is dependent on the direction of the wave vector.

For example the equation

$$u_{tt} = \Delta u - au$$

gives a relation $D^2 = |w|^2 + a$ and it is thus dispersive; the equation $u_{tt} = \nabla \cdot (A \nabla u)$ is an example of an anisotropic equation. The matrix A is symmetric positively definite. The resulting dispersion relation is $D^2 = \langle w, Aw \rangle$.

The discretization itself causes artificial dispersion and anisotropy effects bothering the numerical solution. This property can be seen, for example, by replacing the waveplane solution to the differenced form of a one-dimensional wave equation $u_{tt} = u_{xx}$. As described above, the true solution has no dispersion. Instead, the insertion of a plane wave to the equation

$$\frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{d^2} = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2}$$

gives a relation

$$\frac{e^{-iDd} - 2 + e^{iDd}}{d^2} = \frac{e^{iwh} - 2 + e^{-iwh}}{h^2} \quad \text{or equivalently} \quad \frac{4}{d^2} \sin^2(Dd/2) = \frac{4}{h^2} \sin^2(wh/2)$$

This is called the dispersion relation of a discrete problem and it is easily seen that if the d differs from the h then both the phase and the group velocity depends on the wave vector and they are not equal. The wave concept to were meaningful, the D should be real. This requirement leads to the requirement $d^2 \sin^2(wh/2) \leq h^2$. The previous requirement to be valid for all w the time step d must be smaller than or equal to h. This is called a stability condition of a discrete problem. If this condition is broken there will be exponentially growing components in the numerical solution. Dispersion caused by the numerical solution can be seen as a departure of the original signal from the D'Alambert solution; there will be additional frequent oscillations. Higher dimensional cases will provide more interesting phenomena and they can be analyzed in a similar way.

Concluding words

The field of numerical analysis of partial differential equations is huge. Throughout the text we have considered only few special cases in a rather simple way. To get an idea of this subject it is necessary to take a glance or few to the literature, for example [2] will give a soft introduction.

It is good to keep in mind that difference formulas used in this paper are just certain choices made to approximate the exact relation between the difference and the differential operator: $hD = \log(1 + \Delta)$, where h is the step-size, $\Delta = y_{n+1} - y_n$ the forward difference and D the differential operator. It is also possible to establish relations between forward difference operator and other finite difference operators [2]. Thus, there is a numerable amount of possible choices to approximate this relation but a universal rule that picks out the best one does not exists. Nevertheless, the convergence analysis will show choices that are useless and it may lead to good or even excellent decisions.

Those who are interested in or otherwise forced to employ the advanced numerical analysis may ask help for professors Olavi Nevanlinna and Timo Eirola (Institute of Mathematics).

References

- [1] N.Gershenfeld, Mathematical Modeling
- [2] W.F.Ames, Numerical methods for partial differential equations.
- [3] H.Brezis, Analyse fonctionelle.
- [4] A.R.Mitchell & D.F.Griffiths, The finite difference method in partial differential equations.
- [5] E.Kreyszig, Introductory Functional Analysis with Applications

Related courses

- 1. Mat-1.404 Matematiikan peruskurssi L4, kurssin loppuosa.
- 2. Mat-1.169 Differenssimenetelmät
- 3. Mat-1.175 Numeerinen matriisilaskenta
- 4. Mat-1.140 Funktionaalianalyysin perusteet
- 5. Mat-1.141 Funktionaalianalyysin sovelluksia

Appendix

Semex1.m

```

q=10.0;
symbols=['m:o','b:+';'g--';r: ','k- '];
for k=1:5 ,
    m=2^(k-1)*5-1; h=1/(m+1);

    Ah=(diag((2+q*h^2)*ones(m,1))-...
        diag(ones(m-1,1),1)-diag(ones(m-1,1),-1))/h^2;
    b=zeros(m,1);
    for j=1:m , b(j)=f(j*h);end
    u=[0;Ah\b;0];

    plot(h*(0:m+1),u,symbols(k,:));
    if k==1, hold on; end;
end;
hold off

```

semex2.m

```

a=1;
n1=28;n2=28;h1=1/(n1+1);h2=1/(n2+1);
D1=(-diag(2*ones(n1,1))+diag(ones(n1-1,1),1)+diag(ones(n1-1,1),-1))/h1^2;
D2=(-diag(2*ones(n2,1))+diag(ones(n2-1,1),1)+diag(ones(n2-1,1),-1))/h2^2;
A=-kron(D2,eye(n1))-kron(eye(n2),D1)+a*eye(n1*n2);
b=0*A(:,1);
for i=1:n1
    for j=1:n2

```

```

        b(i+n1*(j-1))=f2([i*h1,j*h2]);
    end
end
u=A\b;U=zeros(n1+2,n2+2);
for j=1:n2
    U(2:(n1+1),j+1)=u((j-1)*n1+1:j*n1);
end
mesh(U)

```

semex3.m

```

n1=28;n2=28;h1=1/(n1+1);h2=1/(n2+1);alfa=0.1;
D1=(-diag(2*ones(n1,1))+diag(ones(n1-1,1),1)+diag(ones(n1-1,1),-1))/h1^2;
D2=(-diag(2*ones(n2,1))+diag(ones(n2-1,1),1)+diag(ones(n2-1,1),-1))/h2^2;
D2(n2,n2)=-1/h2^2;
D=kron(D2,eye(n1))+kron(eye(n2),D1);
b=0*D(:,1);
for i=1:n1
    for j=1:n2
        b(i+n1*(j-1))=f2([i*h1,j*h2]);
    end
end
dt=0.05;nt=40;u=0*b;
R=chol(eye(n1*n2)-0.5*dt*alfa*D);
U=zeros(n1+2,n2+2,nt);
for k=1:nt,w=((u+dt*(0.5*alfa*(D*u)+b))/R)';u=R\w;
for j=1:n2,U(2:(n1+1),j+1,k)=u((j-1)*n1+1:j*n1);
end
U(:,n2+2,k)=U(:,n2+1,k);
end
M=moviein(nt);
for j=1:nt
    mesh(U(:,j));
    axis([1,30,1,30,0,0.08]);
    M(:,j)=getframe;
end
movie(M);

```

semex4.m

```

n1=29; n2=29; h1=1/(n1+1); h2=1/(n2+1); dt=0.01;nt=50;
D1=sparse((-diag(2*ones(n1,1))+diag(ones(n1-1,1),1)+diag(ones(n1-1,1),-1))/h1^2);
D2=sparse((-diag(2*ones(n2,1))+diag(ones(n2-1,1),1)+diag(ones(n2-1,1),-1))/h2^2);
D2(n2,n2)=-1/h2^2;
D=sparse(kron(D2,eye(n1))+kron(eye(n2),D1));
uo=0*D(:,1);
for i=1:n1 , for j=1:n2 ,
end; end;
u=uo+0.5*dt^2*(D*uo); U=zeros(n1+2,n2+2);
for k=1:nt , un=2*u-uo+dt^2*(D*u); uo=u; u=un;

```

```
for j=1:n2 , U(2:(n1+1),j+1)=u((j-1)*n1+1:j*n1); end;
U(:,n2+2)=U(:,n2+1);
mesh(U);
axis([1,n1+2,1,n2+2,-0.5,0.8]); axis off;
drawnow,end;
```

```
function w=f(x)
w=-x+1000*x^7;
```

```
function w=f2(x)
w=0;
if norm(x-[0.2,0.6])<0.1,w=1;end
```

```
function w=f3(x)
w=0; d=norm(x-[0.6,0.5]);
if d<0.2, w=(1+cos(pi*5*d))/2; end
```