UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

A Treatment of Discontinuity in Shock Capturing Finite Difference Methods I.

Mao De-kang

March 1988

CAM Report 88-08

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

1. Introduction

In this article a new and very effective treatment of discontinuity, which can work in all kinds of shock capturing finite difference methods, is introduced. As usual, the scalar conservative law:

$$u_t + f(u)_x = 0 \tag{1.1}$$

with initial value $u(x, 0) = u_0(x)$ is considered first, where f is a twice differentiable, convex function, i.e. $f'' > 0$. We would like to mention here that the treatment presented in this paper can be extended to the system case. In fact, we have already applied it to the system of isentropic flow (refer to [21]).

It is well known that the so-called weak solution of (1.1) is a bounded measurable function $u(x,t)$ which satisfies

$$\int_0^\infty \int_{-\infty}^{\infty} (u\, \phi_t + f(u)\, \phi_x)\, dxdt + \int_{-\infty}^{\infty} u_0(x)\, \phi\, (x, 0)\, dx = 0 \tag{1.2}$$

for all $\phi \in C_0^2((-\infty,\infty)\times[0,\infty))$. (1.2) means that the equality (1.1) is correct only in distribution sense. Hence if $u \in BV$ space, equation (1.1) is allowed to be invalid on a set of measure zero. In fact, such measure zero set often corresponds to the discontinuities of solution.

To approximate equation (1.1), one can use a type of difference schemes

$$L_h\, u_h = 0 \tag{1.3}$$

Corresponding the fact that equation (1.1) can be invalid on a set of measure zero, we can add a grid function R_h to the right side of (1.3), which can be nonzero on some grids. But in order to keep the consistency of discrete equation (1.3) with (1.1), we must make R_h approach zero in distribution sense. We call R_h the artificial terms.

Obviously this is a very common idea, and known by everybody who works on shock capturing methods. In fact, we already have had various kinds of artificial terms,

such as artificial viscosity by Kriess, artificial pressure by Harten, etc. (refer to [22], [11]). But the thing somewhat fresh in this paper is that we construct a very special kind of artificial term, which is different from the former ones. For clarity of approach, let's start with a special case, i.e. a Riemann problem which involves a shock facing to the right. Obviously this the most simple case of discontinuity. We study in this case how to design the artificial term properly, and then generalize the idea to general case.

Riemann problem: The initial value is

$$u_0 = \begin{cases} u_l & \text{if } x \le 0 \\ u_r & \text{if } x > 0 \end{cases}$$

here $u_l > u_r$, and $s = (f(u_l) - f(u_r)) / (u_l - u_r) > 0$. The shock divides the (x,t) plane into two parts. In the left part $u(x,t) \equiv u_l$ and in the right one $u(x,t) \equiv u_r$. Also for convenience we only consider (1.3) to be a general three point scheme

$$u_j^{n+1} = u_j^n + \lambda (h_{j+\frac{1}{2}}^n - h_{j-\frac{1}{2}}^n) \tag{1.4}$$

here $h_{j+\frac{1}{2}}^n = h(u_{j+1}^n u_j^n)$, $h(u,u) = f(u)$, $\lambda = \tau/h$ is the mesh ratio.

Now let us add an artificial term $-p_{j+\frac{1}{2}}/\lambda$ to the numerical flux $h_{j+\frac{1}{2}}^n$ in scheme (1.4). It then is reduced to

$$u_j^{n+1} = u_j^n - \lambda (h_{j+\frac{1}{2}} - h_{j-\frac{1}{2}}^n) + p_{j+\frac{1}{2}}^n - p_{j-\frac{1}{2}}^n \tag{1.5}$$

Because the weak solution of Riemann problem has the properties described above, naturally we consider designing the artificial terms to make the solution also has a similar construction. That is to make the numerical solution u_h^n involve only one discrete shock, which on every time level is a jump occupying two meshes. On its left $u_h^n \equiv u_l$, and on its right $u_h^n \equiv u_r$. It leads us to consider the following problem: suppose on n level the numerical solution u_j^n just has a jump which is located on j^*-1 ~

j^*+1 , then how to construct the artificial terms to make u_j^{n+1} keep the same property, i.e. its jump is still over two meshes.

In order to make the artificial terms local, we only take $p_{j^*-\frac{1}{2}}^{n}$ and $p_{j^*-\frac{1}{2}}^{n}$ to be nonzero. Considering that the shock should face to the right, we have two choices: 1) The jump of numerical solution on $n+1$ level is located on $j^*-1 \sim j^*+1$ (as shown in Fig. 1.1). Write the difference equations at points j^*-1, j^*, j^*+1 respectively, we get

$$\begin{aligned} u_l &= u_l - \lambda \left(h_{j^*-\frac{1}{2}} - f(u_l) \right) + p_{j^*-\frac{1}{2}} \\ u_{j^*}^{n+1} &= u_{j^*}^{n} - \lambda \left(h_{j^*+\frac{1}{2}}^{n} - h_{j^*-\frac{1}{2}}^{n} \right) + p_{j^*+\frac{1}{2}}^{n} - p_{j^*-\frac{1}{2}}^{n} \\ u_r &= u_r - \lambda \left(f(u_r) - h_{j^*+\frac{1}{2}}^{n} \right) - p_{j^*+\frac{1}{2}}^{n} \end{aligned} \tag{1.6}$$

Solve them, we obtain

$$\begin{aligned} u_{j^*}^{n+1} &= u_{j^*}^{n} - \lambda \left(f(u_r) - f(u_l) \right) \\ p_{j^*-\frac{1}{2}}^{n} &= \lambda \left(h_{j^*-\frac{1}{2}}^{n} - f(u_l) \right) \\ p_{j^*\frac{1}{2}}^{n} &= -\lambda \left(f(u_r) - h_{j^*+\frac{1}{2}}^{n} \right) \end{aligned} \tag{1.7}$$

2) The jump on $n+1$ level is located on $j^* \sim j_*+2$ (also as shown in Fig.1.1). Still write down the equations at these points respectively

$$\begin{aligned} u_l &= u_l - \lambda \left(h_{j^*-\frac{1}{2}}^{n} - f(u_l) \right) + p_{j^*-\frac{1}{2}}^{n} \\ u_l &= u_{j^*}^{n} - \lambda \left(h_{j^*+\frac{1}{2}} - h_{j^*-\frac{1}{2}}^{n} \right) + p_{j^*+\frac{1}{2}}^{n} - p_{j^*-\frac{1}{2}}^{n} \\ u_{j^*+1}^{n+1} &= u_r - \lambda \left(f(u_r) - h_{j^*+\frac{1}{2}} \right) - p_{j^*+\frac{1}{2}}^{n} \end{aligned} \tag{1.8}$$

and obtain from above

$$\begin{aligned} u_{j^*+1}^{n+1} &= u_{j^*}^{n} + \left(u_r - u_l \right) - \lambda \left(f(u_r) - f(u_l) \right) \\ p_{j^*-\frac{1}{2}}^{n} &= \lambda \left(h_{j^*+\frac{1}{2}}^{n} - f(u_l) \right) \\ p_{j^*+\frac{1}{2}} &= -u_{j^*}^{n} + u_l + \lambda \left(h_{j^*+\frac{1}{2}}^{n} - f(u_l) \right) \end{aligned} \tag{1.9}$$

The above discussion indicates that there are essentially two different ways to

calculate the artificial terms. We can not use only one way from the beginning of the computation to its end. So in a correct algorithm the two ways must be taken appropriately. Then a question arises: when should the first or second way be employed? Let's consider it at a different angle. Multiplying equation (1.5) by $\phi_j^n h = \phi\,(jh, n\tau)$, summing them by parts with respect to j and n, we arrive at

$$\sum_{n=0}^{n=\infty}\sum_{j=-\infty}^{j=\infty}\left(\frac{\phi_j^{n+1}-\phi_j^n}{\tau}\,u_j^{n+1}+\frac{\phi_j^n-\phi_{j-1}^n}{h}\,h_{j-\frac{1}{2}}^n\right)h\tau+\sum_{j=-\infty}^{j=\infty}h\,\phi_j^0\,u_j^0$$
$$=\sum_{n=0}^{n=\infty}\sum_{j=-\infty}^{j=\infty}\frac{\phi_j^n-\phi_{j-1}^n}{\tau}\,p_{j-\frac{1}{2}}^n\,h\tau \qquad (1.10)$$

When τ, h tend to zero, the left side of (1.10) approach to the left side of (1.2). In order that the approximate solution tend to the weak solution, the right side of (1.10) must approach to zero as h, $\tau \to 0$. Notice that the numerical solution has only one jump on each time level, and assume that it is located on $j^n-1 \sim j^n+1$. Also notice that only two artificial terms, $p_{j^n-\frac{1}{2}}^n$ and $p_{j^n+\frac{1}{2}}^n$ are nonzero on each time level, therefore, the right side of (1.10) is reduced to

$$\sum_{n=0}^{n=\infty}\left(\frac{\phi_{j^n}^n-\phi_{j^n-1}^n}{h}\,p_{j^n-\frac{1}{2}}^n+\frac{\phi_{j^n+1}^n-\phi_{j^n}^n}{h}\,p_{j^n+\frac{1}{2}}^n\right)h\tau\,\lambda^{-1} \qquad (1.11)$$

If $\lambda \geq \delta > 0$, and $p_{j^n-\frac{1}{2}}^n$, $p_{j^n+\frac{1}{2}}^n$ are uniformly bounded with respect to h, τ and n, than (1.11) tends to zero as h, $\tau \to 0$. The main consideration of our method is to keep the the artificial terms $p_{j^n-\frac{1}{2}}$ and $p_{j^n+\frac{1}{2}}^n$ uniformly bounded. From (1.7) and (1.9) we find that they will be uniformly bounded if $u_{j^n}^n$, the value of numerical solution at the middle point of the jump, is uniformly bounded.

We now still assume that the jump on n level is located on $j^*-1 \sim j^*+1$ (i.e. $j^*=j^n$). From (1.7) we find that in the first case, $u_{j^*}^{n+1}$, the value at middle point on $n+1$ level, increases by $\lambda(u_l-u_r)$ over $u_{j^*}^n$. From (1.9) we find that in the second case,

$u_{j^*+1}^{n+1}$, also the value at the middle point on $n+1$ level, decreases by $(1-\lambda s)(u_l-u_r)$ down $u_{j^*}^n$, if CFL- condition, $|\lambda f'(u)| \le 1$, holds. It is then easy to see that in order to keep $u_{j^n}^n$ uniformly bounded we should make the algorithm as follows: when $u_{j^n}^n$ becomes too small, the first way is taken, otherwise the second way. In the algorithm presented in this article, we do this as follows. When

$$u_{j^n}^n < u_l + \lambda\,(\,f(u_l) - f(u_r)\,) \tag{1.12}$$

is true, the first way is taken, otherwise the second way. In doing so, not only the artificial terms are kept uniformly bounded, but also the value of numerical solution at middle point , $u_{j^n}^n$, varies only between u_l and u_r.

In Fig. 6.1 we present such a numerical example computed by this algorithm. Fig. 6.1-b) shows a discrete shock, which is drawn by connecting respectively all the left and right end points successively.

We can treat the case of shock facing to the left in the same way.

For general initial value problem, we can not apply this technique to all the meshes on which u_j^n decreases with respect to j. Because if doing so, we can not make the artificial terms only exist locally. Therefore a positive parameter $\alpha > 0$ is chose, and the technique is only applied to the mesh sections on which the u_j^n decreases, and its jump is greater than α. The mesh sections which is chose to accept the artificial treatment is called *generated section.* The generated section occupies at least two cells. Then the computation on it proceeds as follows. If, for example, there are just two meshes in the section $j^*-1 \sim j^*+1$, then we use the formula (1.5) to compute u_j^{n+1}, with artificial terms $p_{j^*-\frac{1}{2}}^{n}$ $p_{j^*+\frac{1}{2}}^{n}$ defined by (1.7) or (1.9), only in which u_l is replaced by $u_{j^*-1}^n$, and u_r by $u_{j^*+1}^n$.

The exact weak solution may involve many interactions of shocks. This case may also occur in the numerical computation, i.e. discrete shocks meet and merge in some way. The algorithm should handle all these cases. So it is somewhat complicated and involves many techniques. In the following section we shall give an algorithm with such kinds of techniques, however which may be not a good one from the computational viewpoint, but can be regarded as a frame. In section 3 we study some relations, make some comparisons between the present scheme and some other schemes based on Riemann problem, such as Godunov scheme and Glimm scheme. In section 4 we describe how to improve the algorithm given in section 2. More techniques are developed in that section. In section 6 we give sever numerical examples computed by this algorithm.

This is the first one in the series of papers on our work. In the next one in this series some theoretical results on the algorithm such as existence, entropy condition, and convergence, are obtained.

At the same time I was working on this job, professor A. Harten also independently developed a so-called subcell resolution, which was very similar to the treatment studied in this article, although in somewhat different context. Here I would like to express my thanks to professor S. Osher, for he spent some time to make comparison between Harten's method and mine, and found that when the original scheme was UNO or ENO they are just the same. It is a very interesting discovery.

2. Algorithm

In this section a very crude algorithm which employs the treatment of discontinuities described in the previous section is presented. As we mentioned before that from the computational viewpoint this algorithm may be not a good one, especially its treatment of shock interaction. But it is a relatively simple one, and easy to understand and deal with in analysis. Later on, some improvements will be made on it for practical computation. Then it becomes complicated, and one will find that analysis in that case will be a difficult job. Any way, the algorithm presented in this section can be regarded as a frame

Now we present the algorithm in detail.

Choose two positive parameters α_1, and α_2, and $\alpha_1 > \alpha_2$. Because in the initial data the locations of discrete shocks have not been given, or in other words, the candidates that should be artificially treated in computation have not been chosen. So the first thing to do is to defined these candidates, or as we call them, the generated sections on initial level, by parameter α_1 and α_2. It can been done as follows.

1) We define every mesh $j-1 \sim j$ on which $u^0_{j-1} - u^0_j \geq \alpha_1$ as a pre-section.

2) If there are some pre-sections adjacent, i.e. as such: $j_1 \sim j_1+1$, $j_1+1 \sim j_1+2$, ..., $j_1+k-1 \sim j_1+k$, then merge them into one pre-section $j_1 \sim j_1+k$.

3) After the previous two steps, if there are still some pre-sections only of one mesh, we expend them to be pre-sections of two meshes in the following way. Assume $j_1-1 \sim j_1$ to be a pre-section, then the values $u^0_{j_1-2} - u^0_{j_1-1}$, and $u^0_{j_1} - u^0_{j_1+1}$ are checked. If the former one is larger than the later one, then we expend the original pre-section to be $j_1-1 \sim j_1+1$, otherwise $j_1-2 \sim j_1$.

4) After the above expansion if there are still some pre-sections adjacent, merge them again as in the way of step 1).

5) Check the value $u^0_{j_1} - u^0_{j_2}$ on every pre-section. If $u^0_{j_1} - u^0_{j_2} \geq \alpha_2$, then the pre-section will be kept, otherwise it should be cancelled, and also the value of u^0_j on it should be changed in the following way.

$$u^0_j = \begin{cases} u^0_{j_1} & j_1 < j \leq j_1+[(j_2-j_1)/2] \\ u^0_{j_2} & j_1+[(j_2-j_1)/2] < j \leq j_2 \end{cases} \tag{2.1}$$

here [x] takes the maximum integer below x.

The pre-sections produced by the above five steps are the generated sections on initial level. Afterward we shall simply call the first step as generation of pre-sections by α_1, the second and the forth as the mergence, the third as the expansion, and the final one, the cancellation of illegal pre-sections by α_2.

Then we describe how to compute the value of mesh function on $n+1$ level, u^{n+1}_j, and define the relevant generated sections on that level from the mesh function on the previous level, and its generated sections.

step A. Compute the mesh function u^{n+1}_j, and define some pre-sections on this level.

1) To every index j that does not belong to any generated section on n level, we calculate the u^{n+1}_j by the original three point scheme (1.4).

2) Assume that $j_1 \tilde{} j_2$ is a generated section on n level, and $j_2 - j_1 > 2$. In this case u^{n+1}_j is calculated as follows.

$$u^{n+1}_j = \begin{cases} u^n_{j_1} & j_1 \leq j \leq k \\ u^n_{j_2} & k < j \leq j_2 \end{cases} \tag{2.2}$$

here $k=j_1+[(j_2-j_1)]$, and define a pre-section $k-1 \sim k+1$ on $n+1$ level.

3) If the generated section on n level occupies only two grids, i.e. $j_1-1 \sim j_1+1$, then according to four different cases u_j^{n+1} should be computed in the following ways.

If

$$s = (f(u_{j_1+1}^n) - f(u_{j_1-1}^n))/(u_{j_1+1}^n - u_{j_1-1}^n) \le 0$$

then

a) if the mesh function's value at the middle point of generated section satisfies

$$u_{j_1}^n \le u_{j_1+1}^n + \lambda (f(u_{j_1+1}^n) - f(u_{j_1-1}^n)) \tag{2.3}$$

then the artificial terms are defined as

$$\begin{aligned} p_{j_1-\frac{1}{2}}^n &= u_{j_1}^n - u_{j_1+1}^n - \lambda (f(u_{j_1+1}^n) - h_{j_1-\frac{1}{2}}^n) \\ p_{j_1+\frac{1}{2}}^n &= -\lambda(f(u_{j_1+1}^n) - h_{j_1+\frac{1}{2}}^n) \end{aligned} \tag{2.4}$$

b) otherwise the artificial terms are as follows:

$$\begin{aligned} p_{j_1-\frac{1}{2}}^n &= \lambda (h_{j_1-\frac{1}{2}}^n - f(u_{j_1-1}^n)) \\ p_{j_1+\frac{1}{2}} &= -\lambda (f(u_{j_1+1}^n) - h_{j_1+\frac{1}{2}}^n) \end{aligned} \tag{2.5}$$

We add the artificial term $-p_{j-\frac{1}{2}}^n$ to the numerical flux of the original scheme $h_{j-\frac{1}{2}}^n$, and calculate $u_{j_1-1}^{n+1}$, $u_{j_1}^{n+1}$, and $u_{j_1+1}^{n+1}$. by (1.5). If case (a) holds, we have a pre-section $j_1-2 \sim j_1$ on $n+1$ level, otherwise a pre-section $j_1-1 \sim j_1+1$.

If

$$s = (f(u_{j_1+1}^n) - f(u_{j_1-1}^n))/(u_{j_1+1}^n - u_{j_1-1}^n) > 0$$

then

c) if

$$u_{j_1}^n \ge u_{j_1-1}^n + \lambda(f(u_{j_1+1}^n) - f(u_{j_1-1}^n)) \tag{2.6}$$

we define the artificial terms to be

$$\begin{aligned} p^n_{j_1-\frac{1}{2}} &= \lambda(h^n_{j_1-\frac{1}{2}} - f(u^n_{j_1-1})) \\ p^n_{j_1+\frac{1}{2}} &= -u^n_{j_1} + u^n_{j_1-1} + \lambda(h^n_{j_1+\frac{1}{2}} - f(u^n_{j_1-1})) \end{aligned} \tag{2.7}$$

d) otherwise

$$\begin{aligned} p_{j_1-\frac{1}{2}} &= \lambda(h^n_{j_1-\frac{1}{2}} - f(u^n_{j_1-1})) \\ p^n_{j_1-\frac{1}{2}} &= -\lambda(f(u^n_{j_1+1}) - h^n_{j_1+\frac{1}{2}}) \end{aligned} \tag{2.8}$$

Still add $-p^n_{j-\frac{1}{2}}$ to the numerical flux $h^n_{j-\frac{1}{2}}$, and calculate u^{n+1}_j by (1.5). If case (c) takes place, a pre-section $j_1 \sim j_1+2$ on $n+1$ level is given, otherwise a pre-section $j_1-1 \sim j_1+1$.

Step B. Define the generated sections on $n+1$ level.

The previous computation has produced some pre-sections on $n+1$ level, or we say that these pre-sections are inherited from n level. Now we define the generated sections on $n+1$ level by the following steps.

1) First the pre-sections which are adjacent, or even overlapped by each other, should be merged into one pre-section.

2) Cancel the illegal pre-sections by parameter α_2.

On n level, in the regions of non-generated section, the jumps of mesh function u^n_j are relatively weak (bounded by α_1). But after computation, the mesh function in these regions may be not still so even, so smooth. Some jumps stronger than α_1 may occur. So we must define new pre-sections in these areas. Hence:

3) Produce pre-sections by α_1 in these regions.

4) Merge the pre-sections (including the former ones) that are adjacent into one pre-section.

5) If there are still some pre-sections occupying only one cell, expand them to be pre-sections of two cells.

6) Again merge the pre-sections that adjacent to each other into one.

7) Cancel illegal pre-sections by α_2.

The finally formed pre-sections after above seven steps are defined to be the generated sections on $n+1$ level.

This is the entire algorithm.

Remark 2.1 In the present algorithm two parameters α_1, α_2 are employed in generating generated sections. α_1 is used to produce the pre-sections, while α_2 is used to cancel the illegal pre-sections. This case implies that the mesh function u_j^n on newly produced pre-sections at least has a jump greater than α_1; and on cancelled pre-section all its jumps are smaller than α_2. The purpose of such a handling is to prevent the case that discrete shocks occur and disappear too easily. This point is very importment in the theoretical analysis (refer to [23]). But in practical computation we often choose $\alpha_1 = \alpha_2$.

Remark 2.2 If in computation two discrete shocks meet, the algorithm will eventually merge them into one discrete shock. Here we give an example to illustrate this point.

Example. Assume on $n-1$ level there are two generated sections. Each of them belongs to a different discrete shock. Suppose step A. of algorithm produces two adjacent pre-sections on n level (as shown in Fig. 2.1). If no other thing happens, step B. will eventually merge them, and make them to be a generated section occupying four cells on n level. The computation of step A.2) on this generated section will give a

pre-section with two cells on $n+1$ level. Then only one discrete shock will exist afterwards.

Finally we introduce a high resolution treatment of discontinuity. From the previous discussion we see that when generated section has two cells within it , the evolution of the corresponding discrete shock is mainly determined by the value of numerical solution at the middle point of generated section. Hence there must be some relation between this value and the shock position at every time level. We therefore use this value to calculate still further the shock position within the cells. When the generated section has two cells j^n-1 ~ j^n+1, we have following formula

$$spos^n = (j^n h-0.5)h + h(u^n_{j_n} - u^n_{j^n+1}) / (u^n_{j^n-1} - u^n_{j^n+1}) \qquad (2.9)$$

to compute the coordinate of shock position $spos^n$. Obviously when $u^n_{j^n}$ varies from $u^n_{j^n-1}$ to $u^n_{j^n+1}$, $spos^n$ varies from $j^n h - 0.5h$ to $j^n h + 0.5h$. For generated section of more than two cells, a formally generalized one of (2.9) can be used. It is

$$spos^n = (j_1+0.5)h + (\sum_{j=j_1+1}^{j=j_2-1} u^n_j - (j_2-j_1-1)u^n_{j_2}) / (u^n_{j_1} - u^n_{j_2}) \qquad (2.10)$$

here j_1, j_2 are the indexes of the left and right endpoints of the generated section. Clearly when $j_2 - j_1 = 2$, (2.10) is just the same as (2.9). The reasonableness of such a treatment will be studied in the next section.

3. Some analysis of the algorithm

In this section we shall discuss some properties of the present algorithm, study the relations between this scheme and some other ones.

In recent years, a number of shock capturing finite difference schemes based on Riemann problem have been developed. One kind of these schemes is Godunov type scheme. Among them are Godunov scheme, MUSCL scheme, UNO and ENO scheme, etc. (refer to [20], [5], [1], [2], [3], [4]). Another kind is Glimm type scheme (refer to [19]). Also there are a lot of varieties in that type. In order to make thing clear, we would like first to give a short description of these schemes. As for convenience we only consider Godunov scheme and Glimm scheme.

Assume the numerical approximation $u_h^n(x)$ to be a piecewise constant function on n level.

$$u_h^n(x) = \nu_j^n \qquad x_{j-\frac{1}{2}} < x < x_{j+\frac{1}{2}} \tag{3.1}$$

In Godunov scheme, as well as in Glimm scheme, the first thing to do in calculating the approximate solution on $n+1$ level is to solve the IVP:

$$\nu_t + f(\nu)_x = 0, \quad \nu(x,0) = u_h^n(x), \quad n\tau < t \le (n+1)\tau \tag{3.2}$$

Next, some kinds of methods are employed to defined the value of u_h^{n+1} from the solution of this IVP.

The main difference between Godunov scheme and Glimm scheme is located on the definition of u_h^{n+1}. Glimm scheme uses random choice to define the numerical value on the next level, while Godunov scheme uses cell-average.

Obviously $\nu(x,t)$ $(n < t < (n+1)\tau)$ is the exact solution of conservation law. It contains many valuable informations, such as the construction of the solution, the exact

location of the discontinuities, etc. But after the step of random choice (in Glimm scheme), or cell-average (in Godunov scheme), all these useful informations are lost. They do little favor to the following computation. The numerical errors are mainly caused by these two steps.

Many people have noticed this shortcoming and try to overcome it. UNO scheme, ENO scheme, and ENO scheme with subcell resolution developed by A. Harten, S. Osher, B. Engquist, and S.R. Chakravarthy (refer to [1], [2], [3], [4]) reflect such an attempt. All these methods try to recover from the approximate cell-averages, as many as possible, the informations lost; and use them in the late computation, as well as to find the further location of discontinuities. The method of large time step introduced by R.J. LeVeque is an another kind of such attempts. Considering the errors are chiefly caused by the cell-average step in Godunov scheme, LeVeque reduce the number of cell-average steps by taking CFL-number very large.

Here we would like to say that the present treatment is also such an attempt, but of some different idea. The major consideration of it is that while handling the discontinuity, we do not drop these useful informations, but store them in some place. And in computation we use them not only to determine the location of discontinuity, but also to direct the calculation itself. To make the things clear we still start with the example in § 1, i.e. the one of Riemann problem involving a shock facing to right, to illustrate this point.

Suppose that on n level the numerical solution u_j^n has just a jump located on $j_1-1 \sim j_1+1$, on its left $u_j^n \equiv u_l$ and on its right $u_j^n \equiv u_r$. According to the high resolution treatment of discontinuity introduced in the last section, the discrete shock's location is defined on the point

$$spos^n = j_1 h - 0.5h + h(u^n_{j_1} - u_r)/(u_l - u_r) \tag{3.3}$$

Assume at this moment the location of exact shock is just right the same as that of the discrete one, i.e. also $spos^n$ in (3.3), then the thing of interest is how about them on the following level. Because the shock speed $s = (f(u_l) - f(u_r)) / (u_l - u_r)$, so the location of the exact one on n level is always $spos^n + \lambda sh$. Now let's have a look at the discrete shock.

Case 1). If

$$u_{j_1} < u_l + \lambda(f(u_l) - f(u_r)). \tag{3.4}$$

(3.4) tells us that in this case the exact one's location on $n+1$ level is on the left of $j_1 h + 0.5h$ (see Fig.3.1). At that moment the numerical computation makes:

$$u^{n+1}_{j_1} = u^n_{j_1} - \lambda (f(u_l) - f(u_r)) \tag{3.5}$$

and the generated section on $n+1$ level still be $j_1-1 \sim j_1+1$. This indicates that the location of discrete shock on $n+1$ level is also $spos^n + \lambda sh$, just the same as the exact one.

Case 2) If

$$u^n_{j_1} \geq u_l + \lambda (f(u_r) - f(u_l)) \tag{3.6}$$

(3.6) indicates that on $n+1$ level the exact one moves to the right of $j_1 h + 0.5h$ (see Fig.3.2). The computation is different now. $u^{n+1}_{j_1+1}$ is calculated by:

$$u^{n+1}_{j_1+1} = u^n_{j_1} + (u_r - u_l) - \lambda (f(u_r) - f(u_l)) \tag{3.7}$$

and the generated section on $n+1$ level is on $j_1 \sim j_1+2$. But still we can easily find that the locations of the discrete one and the exact one are the same, i.e. $spos^n + \lambda sh$.

The above example shows that the value of numerical solution at the middle point of generated section just contains the information of shock location. This information is not only used in the subcell resolution as in (2.9), but also to direct the computation,

i.e. to check if (3.4) or (3.6) is true, and then decide which calculation step should be taken. Because now there is no cell-average or random choice to cause the error, hence the numerical result in this case is just right the true solution with exact shock curve.

In general initial value problem, the weak solution may involve many shocks. The above analysis tells us that the numerical errors are mainly produced by the computation on the outside part of discontinuities, not the inside part, i.e. the regions without generated sections. If the exact solution is a piecewise smooth function, and the original scheme (1.4) is, for example, of 2th order, we guess that the numerical solution and the corresponding shock curves obtained by present algorithm must have an accuracy of the same order. The numerical example 2 and 3 in § 5 support this conjecture.

The only thing to be destroyed in this algorithm is the consistency, i.e. the truncation error of this scheme does not have zero as its limit everywhere when mesh length h, $\tau \to 0$. One can easily find that the truncation error on generated section is of $O(1)$. But we would like to say that this is not a bad thing, contrary it is a good thing. As we have seen before that the generated section, which always contains a relatively big jump, is regarded as discontinuity in the algorithm. In those mesh sections high order truncation error does not always have good results. Often the case is the higher the order is, the worse the computation. The reason is simple, because the exact weak solution of conservation law does not satisfy the partial differential equation across discontinuity, but the discontinuous condition, i.e. Hugoniot condition. However the artificial terms added to the scheme make it just right to imitate this discontinuity condition. So if there is still some consistence, it must be the consistence between the algorithm and Hugoniot condition in some sense. On the other hand, the exact solution concerned is only a weak one, i.e. the one which only satisfies (1.2), not (1.1).

This fact allows us to do some strong artificial treatment. Exactly say, the grid function R_h added to the right of (1.3) (refer to § 1) can be very strong at some points, even violate the consistency there. The only requirement is that it should tend to 0 when $h \to 0$. In the present discussion this requirement consists of following two limitations on R_h.

1) The R_h is uniformly bounded.

2) The total area of all cells which accept the artificial treatment tends to 0, as grid becomes more and more fine.

Obviously, such two limitations are enough to guarantee that the distribution of 0 is the limit of R_h.

Finally, we would like to say something about our algorithm and Harten's. Though as mentioned in § 1, the ideas of both may be closed to each other, however the implementation of them still have something different. The major difference focuses on the handling of discontinuity. In Harten's algorithm, the computation on each time level is independent. The calculation on n level leaves nothing extra to the next level, except the numerical approximation u_j^{n+1} itself. But in our method, the calculation on n level not only produces u_j^{n+1}, but also some pre-sections which will be considered precedently to be the candidates accepting the artificial treatment on the next level. This has been shown in Step A.2), 3) and Step B.1), 2) in algorithm. Such a handling makes discontinuities relatively coherent and complete (as shown in each examples in § 5). It makes the algorithm have some flavor of shock tracking, or singularity-separating difference method (refer to [17] and [22]).

4. Practical Computation, Some improvements on the Algorithm

As we have said in § 2 that the present algorithm is still very crude. If using it in practical computation, we will find the numerical result may be not very good. The problem exists in the treatment of generated sections of more than two cells. That is step A.2) in the algorithm. The following example will illustrate the case.

Let the initial value be:

$$u_0 = \begin{cases} u_l & x \le a \\ u_m & a < x < b \\ u_r & b \le x \end{cases} \tag{4.1}$$

here $u_l > u_m > u_r$, $a < b$. In this case the true solution of differential equation involves two shocks. They start respectively from $x = a$ and $x = b$, after a period of time they meet somewhere and merge into one shock. If we use the present algorithm to compute the numerical solution with parameters α_1, α_2, and mesh size h small enough, there are also two discrete shocks involved in the approximation. Fig. 4.1 shows the computational situation. Fig. 4.1-a) and -b) are the sketches of shock curves and discrete shocks respectively. In Fig. 4.1-a), the dotted lines represent the shock curves of discrete shock, and the solid ones present that of exact solution. Fig. 4.1-b) assumes that the two discrete shocks meet on n level, and merge into a generated section occupying four cells, then produce a generated section of two cells on $n+1$ level through step A.2).

As shown in Fig. 4.1-a), the distance between the two exact shocks is zero when they meet and merge. But the case of discrete ones is not as such. Let's have a look on n level. If we regard the two pre-sections, from which a generated section of four meshes comes, still as two different generated sections, and calculate the corresponding

$spos_1^n$, $spos_2^n$ of them respectively, we will find the difference between these two values to be at least of h, and at most of $3h$. Here $spos_1^n$ indicates the coordinate of shock position of the left pre-section, and $spos_2^n$ indicates that of the right one. Thus the numerical result is too much different from the true solution. This is just the first problem we have in the algorithm.

Let the computation go on, and then the two discrete shocks merge into one, and produce a generated section of four cells. At that moment the location of discrete shock obtained by (2.10) is still correct. Then step A.2) works. It restricts the generated section on $n+1$ level onto the two middle cells, and also the value of numerical solution on the middle point of it to be u_l. This causes the following locations of discrete shock incorrect (as shown in Fig. 4.1-a)). The error will vary from a half to two or three cells. The practical computation also shows this fact (refer to [21]). If the solution involves many interactions of shocks, the situation will be much worse. This is second problem in computation.

Obviously the algorithm needs some amendments. In this section we first deal with the second problem, and then the first one.

In each case of step A.3), the calculation can be realized by adding some artificial terms on the right side of the original scheme (2.1). Unlike step A.3), all the numbers in step A.2) are defined by force. But we can analyse the step A.2) in the same way as in step A.3), i.e. we still consider that the numerical result of (2.2) is obtained by adding some artificial terms on (1.4). Then the analysis of these artificial terms will help us to reveal the nature of the problem.

Suppose there is a generated section $j_1 \tilde{} j_2$ on $n+1$ (as shown in Fig. 4.2), on its left and right are two constant states u_l and u_r respectively. Considering the

computation is carried out by adding artificial terms $p^n_{j_1-\frac{1}{2}}$, $p^n_{j_1+\frac{1}{2}}$, ..., $p_{j_2-\frac{1}{2}}$, $p_{j_2+\frac{1}{2}}$ on the right side of (2.1), we write down the difference scheme at each of these points,

$$\begin{aligned} u_l &= u^n_j - \lambda(h^n_{j+\frac{1}{2}} - h^n_{j-\frac{1}{2}}) + p^n_{j+\frac{1}{2}} - p^n_{j-\frac{1}{2}} && j_1 \le j < k \\ u_l &= u^n_k - \lambda(h^n_{k+\frac{1}{2}} - h^n_{k-\frac{1}{2}}) + p^n_{k+\frac{1}{2}} - p^n_{k-\frac{1}{2}} && j = k \\ u_r &= u^n_j - \lambda(h^n_{j+\frac{1}{2}} - h^n_{j-\frac{1}{2}}) + p^n_{j+\frac{1}{2}} - p^n_{j-\frac{1}{2}} && k < j \le j_2 \end{aligned} \qquad (4.2)$$

here $k=j_1+[(j_2-j_1)/2]$. Let $p_{j_1-\frac{1}{2}} = 0$, then Eq. (4.2) still has j_2-j_1 unknowns: $p^n_{j_1+\frac{1}{2}}$, $p^n_{j_1+\frac{3}{2}}$, ..., $p^n_{j_2+\frac{1}{2}}$. We can uniquely solve them from (4.2). But at this moment $p_{j_2+\frac{1}{2}}$ may be not zero. To keep the calculation while $j > j_2$ still be carried out by the original three-point scheme (1.4), we must add artificial terms $p^n_{j_2+\frac{1}{2}}$ to every $h^n_{j-\frac{1}{2}}$ ($j > j_2$). This causes the artificial terms not local, i.e. they are nonzero not only in generated section, but also out of it. This is just the reason why the location of discrete shock is incorrect. It is something like the case that an unconservative scheme often gives incorrect shock speed (as mentioned in [16])

Here we give the following amendment to step A.2). To keep the artificial terms local, we take $p^n_{j_2+\frac{1}{2}} = 0$ in Eq. (4.2), but at the meantime let u^{n+1}_k in the kth equation be free. Now Eq. (4.2) still has j_2-j_1 unknown; and we can solve them uniquely, and then artificialterms exist only locally.

Moreover, to restrict the generated section onto $n+1$ level over the two middle cells is also considered unreasonable. We wish to have several choices of k, such as in the case of step A.3), i.e. k would be any index between j_1 and j_2. In this article we define k as follows. If $s \le 0$, we take k to be j_1, j_{+1}, ..., j_2-1, and solve the corresponding system (4.2) respectively. calculate each difference $|u^{n+1}_k - \frac{1}{2}(u_l+u_r)|$,

then k is defined to be the index which makes its corresponding difference to be the minimum among all these differences. When $s > 0$ we define k in the same way as in the case of $s < 0$; except the index j_1 is replaced by j_2. Such a definition of k makes the mesh function u_j^{n+1} have less oscillation.

Now let's try to solve the first problem. Obviously the first problem is caused by the too early mergence of generated sections. So step B. needs some revision to delay the mergence of generated sections. Let's solve this problem step by step.

1. The first amendment is as follows. When two pre-sections on n level are adjacent (as shown in Fig. 4.1-b)), the algorithm now will not merge them into one, but still regard them as two independent generated sections. Calculate the artificial terms on them by step A.3), and apply them to the computation. This amendment can delay the mergence a little bit. However, one would like to know how about the computational result now. The following examlpes will shows us that the result on $n+1$ level is just as that when the two generated sections exist independently. There is no interference between them.

As shown in Fig. 4.3, there are two adjacent generated sections $j_1-2 \sim j_1$, and $j_1 \sim j_1+2$ on $n+1$ level. Assume that on their left and right are u_l, and u_r respectively, and $u_{j_1}^n = u_m$. If step A.3.a) in algorithm works on the left generated section, and step A.3.d) works on the right one, then we get:

$$\begin{aligned} p_{j_1-\frac{3}{2}}^n &= u_{j_1-1}^n - u_m - \lambda(f(u_m) - h_{j_1-\frac{3}{2}}^n) \\ p_{j_1-\frac{1}{2}}^n &= -\lambda(f(u_m) - h_{j_2-\frac{1}{2}}^n) \end{aligned} \tag{4.3}$$

and

$$\begin{aligned} p_{j_1+\frac{1}{2}}^n &= \lambda(h_{j_2+\frac{1}{2}}^n - f(u_m)) \\ p_{j_1+\frac{3}{2}} &= -\lambda(f(u_r) - h_{j_1+\frac{3}{2}}^n) \end{aligned} \tag{4.4}$$

It is easy to obtain

$$\begin{aligned}
u_{j_1-3}^{n+1} &= u_l \\
u_{j_1-2}^{n+1} &= u_{j_1-1}^{n} + \lambda(f(u_l) - f(u_r)) \\
u_{j_1-1}^{n+1} &= u_m \\
u_{j_1+1}^{n+1} &= u_{j_1+1}^{n} - \lambda (f(u_r) - f(u_m)) \\
u_{j_2+2}^{n+1} &= u_r \\
u_{j_1}^{n+1} &= u_m - \lambda(h_{j_1-\frac{1}{2}}^{n} - h_{j_1-\frac{1}{2}}^{n}) + p_{j_1+\frac{1}{2}}^{n} - p_{j_1-\frac{1}{2}}^{n} = u_m
\end{aligned} \tag{4.5}$$

Also if the computation on generated section $j_2-2 \sim j_1$ is carried on according to step A.3.c), and on $j_1 \sim j_1+2$ according to step A.3.b), we can obtain:

$$\begin{aligned}
p_{j_1-\frac{3}{2}}^{n} &= \lambda(h_{j_1-\frac{3}{2}}^{n} - f(u_l)) \\
p_{j_1-\frac{1}{2}}^{n} &= -u_{j_1-1}^{n} + u_l + \lambda(h_{j_1-\frac{1}{2}}^{n} - f(u_l))
\end{aligned} \tag{4.6}$$

and

$$\begin{aligned}
p_{j_1+\frac{1}{2}}^{n} &= \lambda(h_{j_1+\frac{1}{2}}^{n} - f(u_m)) \\
p_{j_1+\frac{3}{2}}^{n} &= -\lambda(f(u_r) - h_{j_1+\frac{3}{2}}^{n})
\end{aligned} \tag{4.7}$$

and still easy to get $u_{j_1-1}^{n+1} = u_l$, $u_{j_1+2}^{n+1} = u_r$, and

$$\begin{aligned}
u_{j_1}^{n+1} &= u_m - \lambda(h_{j_2+\frac{1}{2}}^{n} - h_{j_1-\frac{1}{2}}^{n}) + p_{j_1+\frac{1}{2}}^{n} - p_{j_1-\frac{1}{2}}^{n} \\
&= u_{j_1-1}^{n} + u_m - u_l - \lambda(f(u_m) - f(u_l))
\end{aligned} \tag{4.8}$$

When the left generated section on n level exits alone, the result of (4.8) is just the value of numerical solution on the middle point of the corresponding generated section on $n+1$ level. Also

$$u_{j_1+1}^{n+1} = u_{j_1+1}^{n} - \lambda(f(u_r) - f(u_m)). \tag{4.9}$$

This is also the value on middle point on $n+1$ level when the right generated section exist alone. Hence the case is also similar to that when two generated sections exist independently, except the absence of middle state u_m.

To all other cases, the same conclusion also can be obtained, especially when generated section has more than two meshes, and even more than two generated sections adjacent are involved.

Though this revision makes the computation a little bit better, however the numerical results are often still not very good. Let's observe the example presented in Fig. 4.1. As for convenience we mark the left and right discrete shock by the discrete shock 1 and discrete shock 2 respectively. Now we continue our computation, and assume it proceeds as shown in Fig. 4.4. On $n+2$ level the computation in the left generated section takes step A.3.c), while on the right one takes step A.3.d). Therefore there will be two overlapped pre-sections which have one common cell on $n+3$ level. According to the algorithm, step B. should combine them into one generated section. But if we compare $spos_1^{n+3}$, and $spos_2^{n+3}$ calculated by (2.9), we will always find still $spos_1^{n+3} < spos_2^{n+3}$, and sometime the difference between them may be of $2h$. Obviously it indicates that the mergence still comes too early. Therefore the second amendment is that the two pre-sections in this case should not be merged, but regarded as two independent generated sections again.

As we see, there are two grid points within these two generated sections. The values of numerical solution on these two points are taken to be the middle values of these two generated sections respectively (the above second example supports the reasonableness of such a consideration). There still needs a middle state, that is the right state of generated section 1, and the left state of generated section 2. In general, this middle state should be defined in different ways according to different cases. Hence when the algorithm or its corresponding computational program is created, all possible cases should be considered. This middle state does not occur in the numerical solution

itself. Therefore in the program we shall use a special unit to store it.

Now let's considered how to proceed our computation on these two overlapped ones, i.e. how to construct the artificial terms. Assume that the two generated sections are $j_1-1 \sim j_1+1$ and $j_1 \sim j_1+2$ on n level, $j_1 \sim j_1+1$ is their common cell. The left, right, and middle states are u_l, u_r, and u_m respectively. Considering each of these two generated sections alone, we can calculate their artificial terms as $p^{n,1}_{j_1-\frac{1}{2}}$, $p^{n,1}_{j_1+\frac{1}{2}}$, $p^{n,2}_{j_1+\frac{1}{2}}$, and $p^{n,2}_{j_1+\frac{3}{2}}$. Here $p^{n,1}_{j_1+\frac{1}{2}}$ involves $h^{n,1}_{j_1+\frac{1}{2}} = h(u^n_{j_1}, u_m)$, and $p^{n,2}_{j_1+\frac{1}{2}}$ involves $h_{j_1+\frac{1}{2}} = h(u_m, u^n_{j_1+1})$. Now we replace both $h^{n,1}_{j_1+\frac{1}{2}}$ in $p^{n,1}_{j_1+\frac{1}{2}}$, and $h^{n,2}_{j_1+\frac{1}{2}}$ in $p^{n,2}_{j_1+\frac{1}{2}}$ by $h^n_{j_1+\frac{1}{2}} = h(u^n_{j_1}, u^n_{j_1+1})$, then obtain the corresponding numbers $\tilde{p}^{n,1}_{j_1+\frac{1}{2}}$ and $\tilde{p}^{n,2}_{j_1+\frac{1}{2}}$. Finally we take

$$\begin{aligned} p^n_{j_1-\frac{1}{2}} &= p^{n,1}_{j_1-\frac{1}{2}} \\ p^n_{j_1+\frac{1}{2}} &= \tilde{p}^{n,1}_{j_1+\frac{1}{2}} + \tilde{p}^{n,2}_{j_1+\frac{1}{2}} + \lambda(f(u_m) - h^n_{j_1+\frac{1}{2}}) \\ p^n_{j_1+\frac{3}{2}} &= p^{n,2}_{j_1+\frac{3}{2}}. \end{aligned} \tag{4.10}$$

Taking artificial terms in this way will make the computation just act as that when two generated sections exist independently. No interference will be between them. For example, if observed alone, both the computations on $j_1-1 \sim j_1+1$ and $j_1 \sim j_1+2$ should act as step A.3.a), hence it is easy to obtain:

$$\begin{aligned} p^n_{j_1-\frac{1}{2}} &= u^n_{j_1} - u_m - \lambda(f(u_m) - h^n_{j_1-\frac{1}{2}}) \\ p^{n,1}_{j_1+\frac{1}{2}} &= -\lambda(f(u_m) - h^{n,1}_{j_1+\frac{1}{2}}) \end{aligned} \tag{4.11}$$

and

$$\begin{aligned} p^{n,2}_{j_1+\frac{1}{2}} &= u^n_{j_1+1} - u_r - \lambda(f(u_r) - h^{n,2}_{j_1+\frac{1}{2}}) \\ p^n_{j_1+\frac{3}{2}} &= -\lambda(f(u_r) - h^n_{j_1+\frac{3}{2}}) \end{aligned} \tag{4.12}$$

Then:

$$\begin{aligned} \tilde{p}^{n,1}_{j_1+\frac{1}{2}} &= -\lambda(f(u_m) - h^n_{j_1+\frac{1}{2}}) \\ \tilde{p}^{n,2}_{j_1+\frac{1}{2}} &= u^n_{j_1+1} - u_r - \lambda(f(u_r) - h^n_{j_1+\frac{1}{2}}) \end{aligned} \tag{4.13}$$

Finally,

$$\begin{aligned} p^n_{j_1+\frac{1}{2}} &= \tilde{p}^{n,1}_{j_1+\frac{1}{2}} + \tilde{p}^{n,2}_{j_1+\frac{1}{2}} + \lambda(f(u_m) - h^n_{j_1+\frac{1}{2}}) \\ &= u^n_{j_1+1} - u_r - \lambda(f(u_m) - h^n_{j_1+\frac{1}{2}}) \end{aligned} \tag{4.14}$$

Adding them onto the right side of the original scheme (1.4), one can easily obtain:

$$\begin{aligned} u^{n+1}_{j_2-2} &= u_l \\ u^{n+1}_{j_1-1} &= u^n_{j_1} + (u_l - u_m) - \lambda(f(u_m) - f(u_l)) \\ u^{n+1}_{j_1} &= u^n_{j_1+1} + (u_m - u_r) - \lambda(f(u_r) - f(u_m)) \\ u^{n+1}_{j_1+1} &= u_r \end{aligned} \tag{4.15}$$

This is just the conclusion. It is also easy to verify that the similar conclusions hold in all the other cases, especially when more than two generated sections are involved.

This amendment can be generalized to the case involving generated section of more than two meshes. In fact such a generalization has been employed in each numerical example presented in § 5.

After the above steps of revision, the computation has been improved a lot. But we are not satisfied yet. If we continue the the computation in the original example, and assume that on $n+3$ level the artificial terms on generated section 1 are calculated by step A.3.d), and on generated 2 by step A.3.a), then on $n = 4$ level there will be two pre-sections, which overlapped entirely with each other. According to the algorithm, step B. should merge them. But if we analyse the situation in the same way as before by comparing $spos_1^{n+4}$, and $spos_2^{n+4}$, we will find sometime the later one is still bigger than the former one. In the worst case the difference between them may be of

h. Obviously the mergence of them is still a little bit early.

Our third amendment is not to merge the pre-section in this case, but still regard them as two independent generated sections. In fact this is a further implementation of the previous idea. On these two overlapped generated sections, not only the middle states, but also the middle values of them do not appear in the numerical solution. So in the computational program we must use several special units to store them. In the present example, the two middle values on $n+4$ level should be calculated respectively as each of the two generated sections on $n+3$ level exists alone. In different cases the middle values should be taken in different way. The algorithm should handle all these cases.

The artificial terms on these two overlapped generated sections should be constructed in the following way. Considering each of them alone, we design the artificial terms to be $p^{n,1}_{j_1-\frac{1}{2}}$, $p^{n,1}_{j_1+\frac{1}{2}}$, $p^{n,2}_{j_1-\frac{1}{2}}$, and $p^{n,2}_{j_1+\frac{1}{2}}$. Replace the u_m in $p^{n,1}_{j_1-\frac{1}{2}}$ by u_r, and in $p^{n,2}_{j_1+\frac{1}{2}}$ by u_l. Then they are just the artificial terms needed.

This amendment can be extended to the case involving more than two overlaped generated sections. The practical computation shows that after these three revisions the numerical result will be very good. This is just illustrated by each examples in § 5.

Finally, in the practical computation the calculation of (2.1) is always given up, for such a change of the value of numerical solution will seriously violate the conservation of the scheme.

5. Numerical Experiments

In this section several numerical examples computed by the algorithm described in the previous sections are presented. Here we take the original scheme to be the one in [9]. That is a Lax- Wendroff scheme with a numerical viscosity, and is of 2th order. Because the present treatment only be focused on shock and contact discontinuity, so if we employ only Lax-Wendroff scheme in computation, there will be some trouble near rarefaction wave. But the numerical viscosity, just as analysed in [9], will eliminate the trouble, and make the numerical solution in the corresponding part somewhat good. That is the reason why we choose such a scheme as a candidate.

The partial differential equation is taken to be

$$u_t + (\frac{1}{2}u^2)_x = 0$$

Example 1. The initial value is

$$u_0(x) = \begin{cases} 2 & x < 0.205 \\ 0 & x \geq 0.205 \end{cases}$$

This problem involves a shock with speed of 1. Let $\alpha_1 = \alpha_2 = 0.5$, $h = 0.01$, $\lambda = 0.45$. The numerical result is presented in Fig. 5.1. Fig. 5.1-a) shows a shock curve, which was drawn by connecting all the $spos^n$ on every time level successively. Fig. 5.1-b) shows the discrete discrete shock, which was drawn by connecting successively all the left and right endpoints of generated sections respectively. Fig. 5.1-c) shows the numerical solution at time $t = 0.45$. Fig. 5.1 indicates that the numerical solution obtained is almost the exact solution.

Afterward, in each example Fig. a) always stands for shock curve, while Fig. b) and c) always stand for discrete shock and numerical solution.

Example 2. The initial value is

$$u_0(x) = \begin{cases} 1 & x < 0.185 \\ -1 & 0.185 \le x < 0.2 \\ \dfrac{x - 0.5}{0.3} & 0.3 \le x < 0.8 \\ 1 & 0.8 \le x \le 1 \end{cases}$$

This problem involves a shock with speed of 0, which starts from $x = 0.185$, and a rarefaction wave with its center at $x = 0.5$. The interaction between them eliminate part of the rarefaction one, and weaken the shock. Take $\alpha_1 = \alpha_2 = 0.4$, $h = 0.01$, $\lambda = 0.9$. The numerical result is presented in Fig. 5.2. The comparison of shock locations between the approximate and exact solution on each time level is presented in Table 5.1. AS stands for the approximate solution, while TS stands for the exact one. Table 1. shows that the error is about 10^{-4}. Noticing $h = 0.01$, one can believe that the error is of $O(h^2)$.

Example 3.

$$u_0 = \begin{cases} 2 & x < 0.185 \\ -1 & 0.185 \le x < 0.2 \\ \dfrac{x - 0.5}{0.3} & 0.2 \le x < 0.8 \\ 1 & 0.8 \le x < 0.815 \\ -2 & 0.815 \le x \end{cases}$$

There are two shocks in this problem. One starts from $x = 0.185$ with speed of $\frac{1}{2}$, another starts from $x = 0.815$ with speed of $-\frac{1}{2}$. In addition, there is a rarefaction wave with it's center at $x = 0.5$. After a period of time the rarefaction wave has been completely eliminated. Let $\alpha_1 = \alpha_2 = 1$, $h = 0.01$, $\lambda = 0.45$. The numerical result is shown in Fig. 5.3. The numerical comparison tells us that the error between the shock

curves of numerical solution and of exact solution is also of $O(h^2)$.

Example 4. The initial value is

$$u_0 = \begin{cases} 2 & x < 0.205 \\ 1 & 0.205 \le x < 0.405 \\ 0 & 0.405 \le x < 0.605 \\ -1 & 0.605 \le x < 0.805 \\ -2 & 0.805 \le x \end{cases}$$

In this problem four shocks with speeds of $\frac{3}{2}$, $\frac{1}{2}$, $-\frac{1}{2}$, and $-\frac{3}{2}$ start from 0.205, 0.405, 0.605, 0.805 respectively. They finally come into one at time $t = 0.2$. In computation α_1 and α_2 are taken to be 0.5, $h = 0.01$, $\lambda = 0.45$. The numerical result indicates that the approximate solution is almost the exact one.

Example 5. The initial value was given by the program 2 in appendix, there $u(100)$ to $u(200)$ are the values at 101 grid points among [0, 1]. This problem involves 10 shocks, which are arranged in a uniformly distributed form when $t = 0$, and merge into one shock when $t = 0.1$. Let $\alpha_1 = \alpha_2 = 0.5$, $h = 0.01$, $\lambda = 0.12$. Fig 5.5 shows the numerical result, and it also almost the exact solution.

Example 6. Initial value is given by the second program in appendix. $u(100)$ to $u(200)$ are the values at each point on initial level. This one is the same as the above one, except 20 shocks are involved now. Let $\alpha_1 = \alpha_2 = 0.5$, $h = 0.01$, $\lambda = 0.065$. The computational result is also very good (see Fig 5.6).

All example 4 to example 6 deal with the interactions of many shocks. Especially in example 6, after 50 steps of computation the discrete shocks begin to pack and crowd, and a lot of adjacencies and overlaps among them take place.

Example 7. The initial value is given by the third program in appendix, there

$u(100)$ to $u(200)$ are the values at each point on initial level. 10 shocks are involved. At $t = 0$ they are arranged in such form: the distance between every two adjacent ones increases from left to right. When t increases they are merged one after another, and finally become a single shock with speed of 0. Take $\alpha_1 = \alpha_2 = 0.4$, $h = 0.01$, $\lambda = 0.135$. The numerical result is shown in Fig. 5.7, and also almost the exact solution.

Example 8. The initial value is defined by program 4 in appendix. There $u(100)$ to $u(200)$ are the values at each point among [0, 1]. This example is just the same as the above one, except there are 18 shocks involved. The numerical result is very good (shown in Fig. 5.8).

Also there are many overlaps and adjacencies occur in the example 7 and 8.

In the present algorithm, in addition to h, τ, there is another parameter $\alpha_1 = \alpha_2$. The interesting question is what influence this parameter has on computation. The following example shows this point.

Example 9. The initial value is still given as that in example 8. Let $h = 0.01$, $\lambda = 0.07$, but $\alpha_1 = \alpha_2$ is taken to be 0.85, 0.75, 0.65, 0.6, 0.55, 0.5 respectively. Because the computational results in this case are completely unforeseen, so this problem also tests if the numerical result changes greatly when α varies., or the insensibility of the algorithm. Here we only present the figures of shock curves in each case (all the figures of numerical solution are just the same as in Fig. 5.8). Fig. 5.9 Shows that when $\alpha_1 = \alpha_2$ decreases down to 0.5, the corresponding approximation is closed to the exact one. It also indicates that the algorithm is some what insensible.

Finally, I would like to express a lot of thanks to my colleagues Mr. Li Zhi-liang and Mr. Xiang Guo-di for their assistance in computing all the examples in § 6.

Appendix

program 1.

```
10   DIM U(300)
20   OPEN "DATA.U" FOR OUTPUT AS#1
30   FOR I=1 TO 102
40   U(I)=5 : WRITE#1,U(I)
50   NEXT
60   K=103
70   FOR I=0 TO 9
80   KK=10*I+105
90   FOR J=K TO KK-1
100  U(J)=5-I : WRITE#1,U(J)
110  NEXT
120  K=KK+1
130  U(KK)=4.5-I : WRITE#1,U(KK)
140  NEXT
150  FOR I=K TO 300
160  U(I)=-5 : WRITE#1,U(I)
170  NEXT
180  CLOSE
190  END
```

program 2.

```
10   DIM U(300)
20   OPEN "DATA.U" FOR OUTPUT AS#1
30   FOR I=1 TO 101
40   U(I)=10 : WRITE#1,U(I)
50   NEXT
60   K=102
70   FOR I=0 TO 19
80   KK=5*I+103
90   FOR J=K TO KK-1
100  U(J)=10-I : WRITE#1,U(J)
110  NEXT
120  K=KK+1
130  U(KK)=9.5-I : WRITE#1,U(KK)
140  NEXT
150  FOR I=K TO 300
160  U(I)=-10 : WRITE#1,U(I)
170  NEXT
180  CLOSE
190  END
```

```
10   DIM U(300)
20   OPEN "DATA.U" FOR OUTPUT AS#1
30   FOR I=1 TO 101
40   U(I)=10 : WRITE#1,U(I)
50   NEXT
60   K=102
70   FOR I=0 TO 19
80   KK=5*I+103
90   FOR J=K TO KK-1
100  U(J)=10-I : WRITE#1,U(J)
110  NEXT
120  K=KK+1
130  U(KK)=9.5-I : WRITE#1,U(KK)
140  NEXT
150  FOR I=K TO 300
160  U(I)=-10 : WRITE#1,U(I)
170  NEXT
180  CLOSE
190  END
```

```
10   DIM U(300)
20   OPEN "DATA.U" FOR OUTPUT AS#1
30   FOR I=1 TO 100
40   U(I)=9 : WRITE#1,U(I)
50   NEXT
60   K=101
70   FOR I=0 TO 17
80   KK=3*I+I*(I-1)/6+102
90   D=KK-INT(KK)
100  IF D>.5 THEN 120
110  KK=INT(KK) : GOTO 130
120  D=D-1 : KK=INT(KK)+1
130  FOR J=K TO KK-1
140  U(J)=9-I : WRITE#1,U(J)
150  NEXT
160  U(KK)=D+8.5-I : WRITE#1,U(KK)
170  K=KK+1
180  NEXT
190  FOR I=K TO 300
200 U(I)=-9 : WRITE#1,U(I)
210  NEXT
220  CLOSE
230  END
```

References

[1] A. Harten; ENO scheme with subcell resolution. manuscript.

[2] A. Harten, B. Engquist, S. Osher, S.R. Chakravarthy; Uniformly high order accurate essentially non-oscillatory schemes, III. J. Comp. Phys. Vol. 71 (1987) 231-303.

[3] A. Harten & S. Osher; Uniformly high-order accurate non-oscillatory schemes, I. SINUM, Vol. 24 (1987) 279-309

[4] Chi-Wang Shu & S. Osher; Efficient implementation of essentially non-oscillatory shock capturing schemes. ICASE report No. 87-33 (1987).

[5] S. Osher; Convergence of generalized MUSCL schemes. SINUM. Vol. 22 (1985) 947-961.

[6] A. Harten; High resolution schemes for hyperbolic conservation laws. J. Comp. Phys. Vol. 49 (1983) 357-393.

[7] S. Osher & S.R. Chakravarthy; High resolution scheme and the entropy condition. SIAM J. Numer. Anal. Vol. 21 (1984) 955-984.

[8] S. Osher; Riemann solver, the entropy condition, and difference approximations. SIAM J. Numer. Anal. Vol. 21 (1984) 227-235.

[9] A. Majda & S. Osher; Numerical viscosity and the entropy condition. Comm. Pure Appl. Math. Vol. 32 (1979) 797-838.

[10] A. Harten, J.M. Hyman, & P.D. Lax; On finite difference approximations and entropy conditions for shocks. Comm. Pure Appl. Math. Vol. 29 (1976) 297-322.

[11] A. Harten; The method of artificial compression. AEC Research and development report c00-3077-50, New York University 1974.

[12] R. Sanders; On convergence of monotone finite difference schemes with variable spatial differencing. Math. Comp. Vol. 18 (1983).

[13] R.J. LeVeque; A Large time step generalization of Goduvov scheme for system of conservation laws. SINUM Vol. 22 (1985) 1051-1073.

[14] R.J. LeVque; Large time step shock-capturing technique for scalar conservation laws. SINUM Vol. 19 (1982) 1091-1109.

[15] P.K. Sweby; High resolution scheme using flux limiters for hyperbolic conservation laws. SIAM J. Numer. Anal. Vol. 21 (1984) 995-1011.

[16] B. Engquist & S. Osher; One-side difference approximations for nonlinear conservation laws. Math. Comp. Vol. 36 (1981) 321-351.

[17] Wu Xiong Hua & Zhu You Lan; A scheme of the singularity-separating method for the nonconvex problem. Comput. & Fluid, Vol. 13, No. 4 (1985) 473-484.

[18] B. van Leer; Towards the ultimate conservative difference scheme. J. Comp. Phys. Vol. 14 (1974) 361-370.

[19] J. Glimm; Solution in the large for nonlinear hyperbolic system of equations. Comm. Pure Appl. Math. Vol. 18 (1965) 697-715.

[20] Godunov; A difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. Mct. sb. 47 (1959) 271-290.

[21] Mao De-kang; A finite difference scheme for shock calculation. J. computational Math. No. 3 (1985) 265-282 (in Chinese).

[22] P. Charrier & B. Tessieras; On front-tracking methods applied to hyperbolic system of nonlinear conservation laws. SIAM J. Numer. Anal. Vol. 23 (1986).

[23] Mao De-kang; A treatment of discontinuity in shock capturing finite difference methods, II. (in preparing).

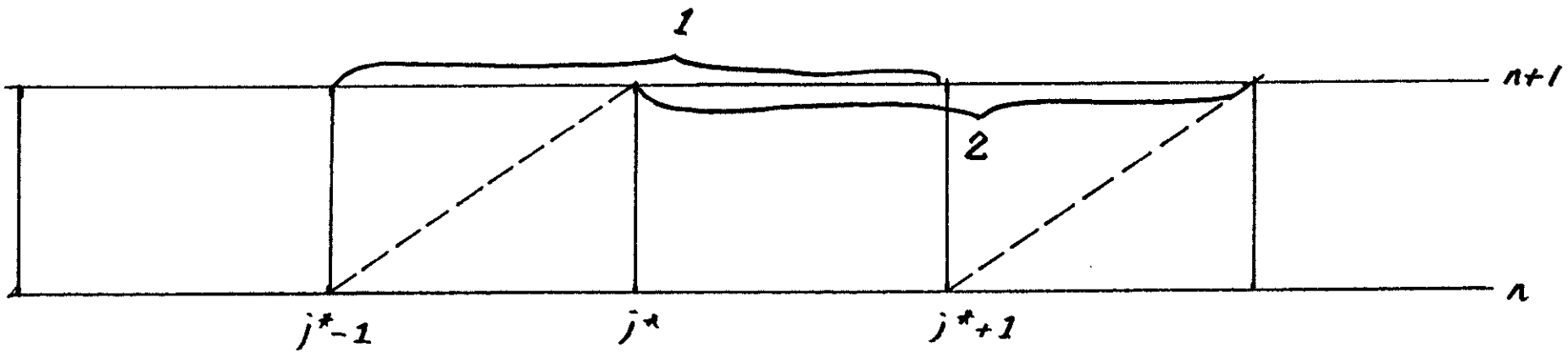


Fig 1.1

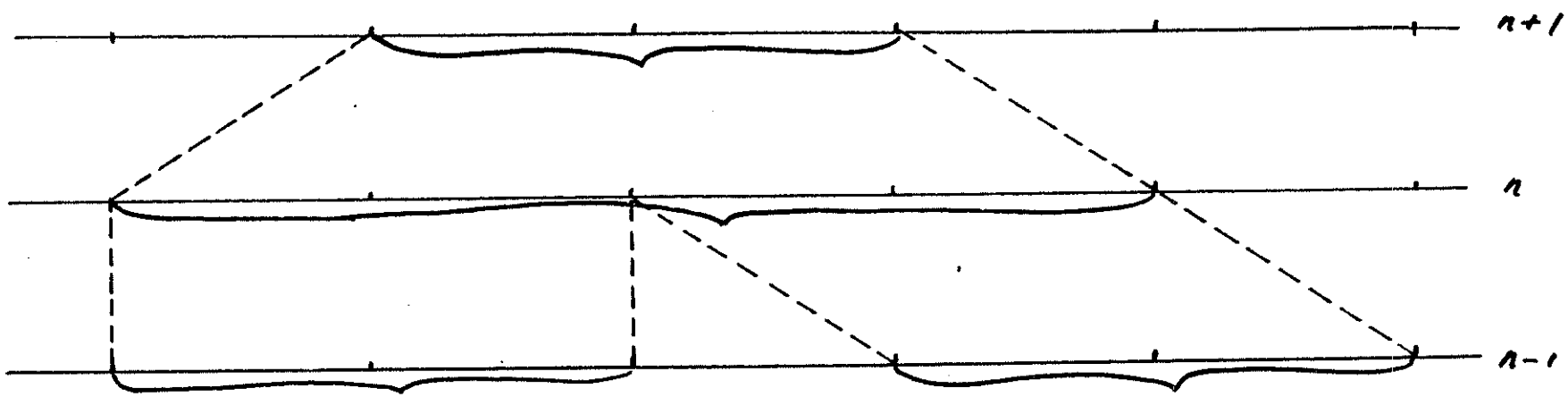


Fig 2.1

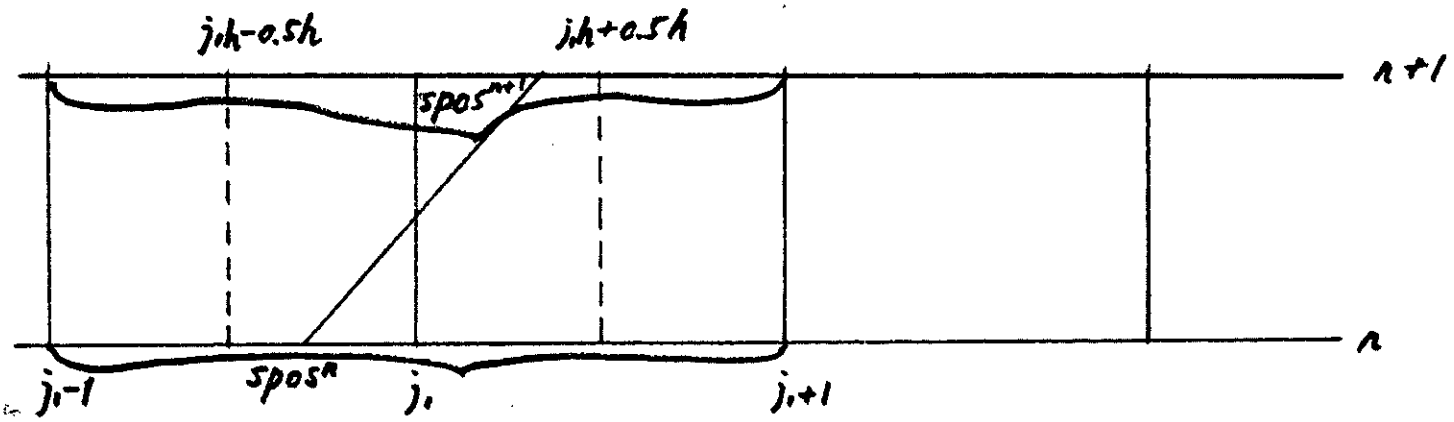


Fig. 3.1

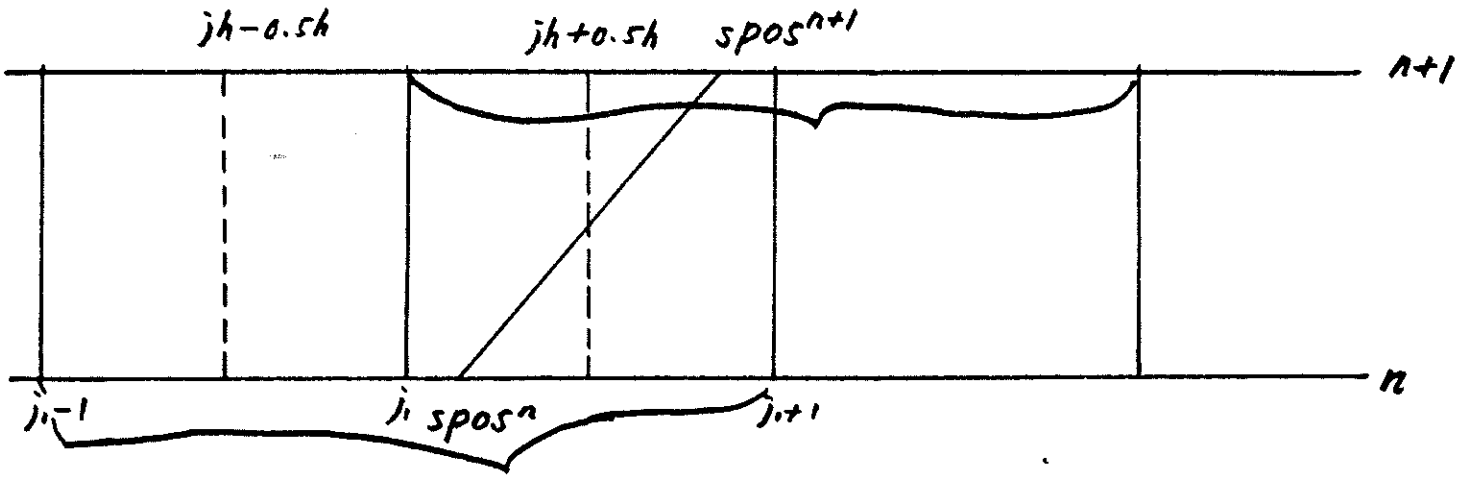
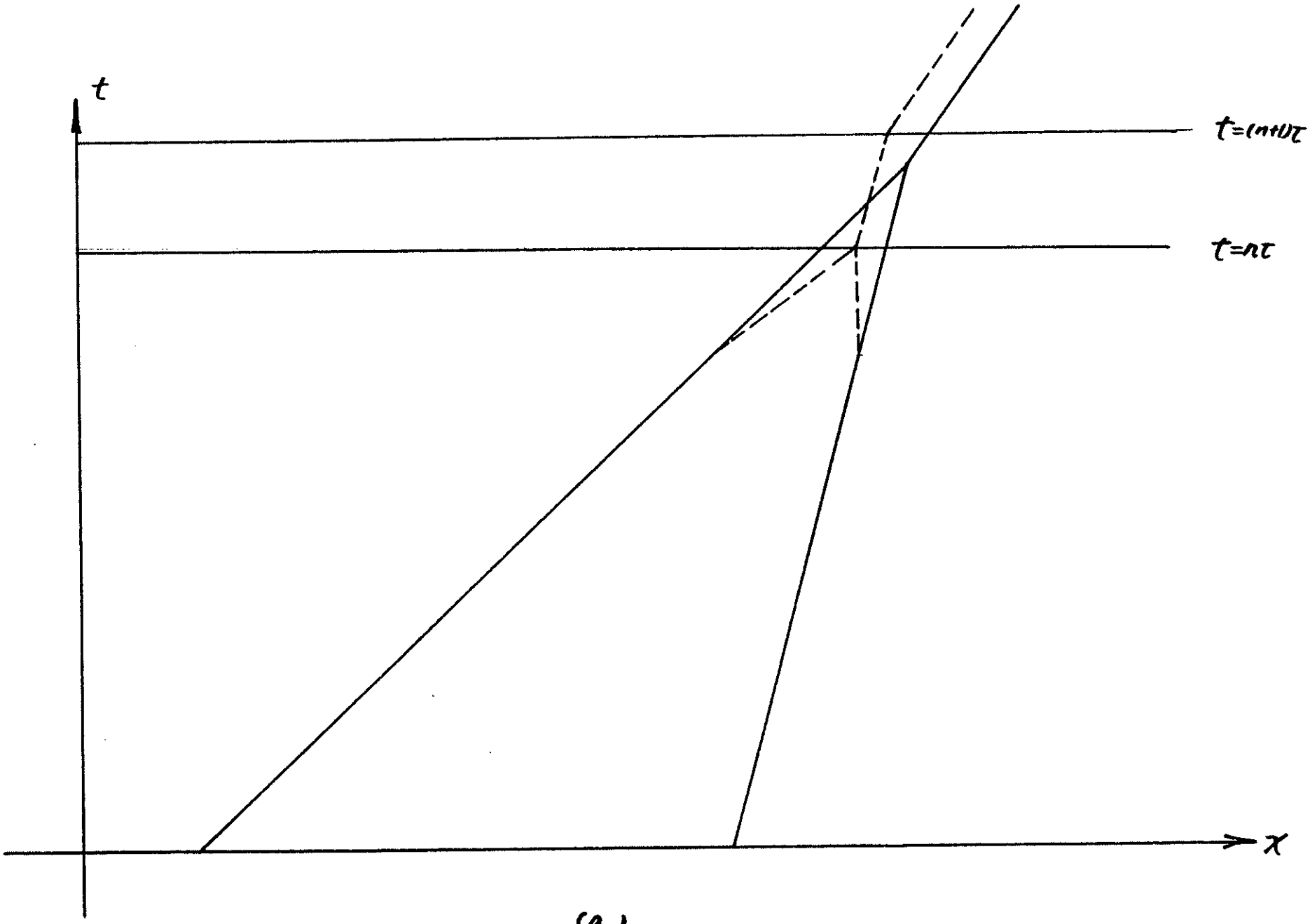
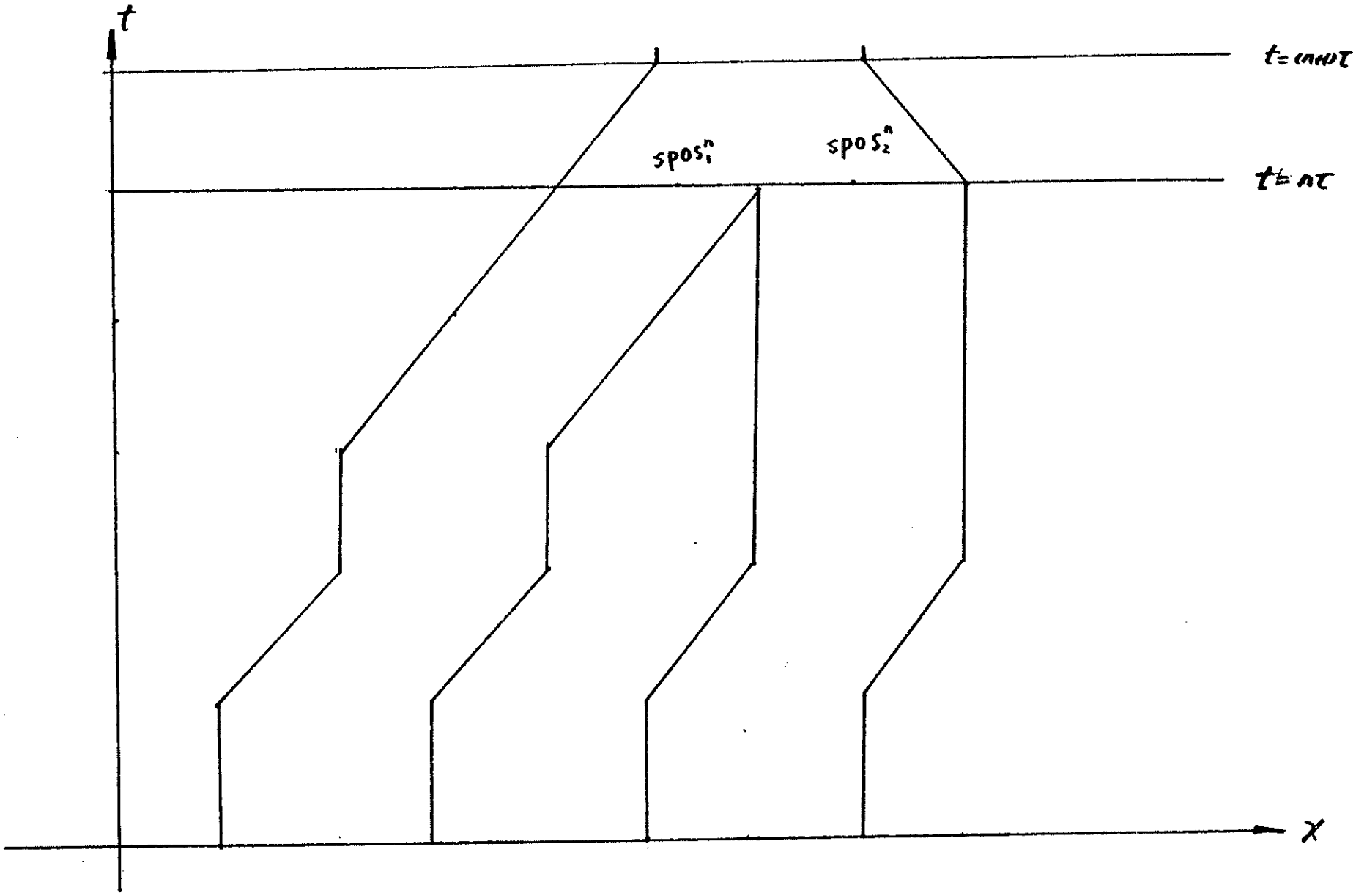


Fig. 3.2



(a)

Fig. 4.1



(b)

Fig. 4.1

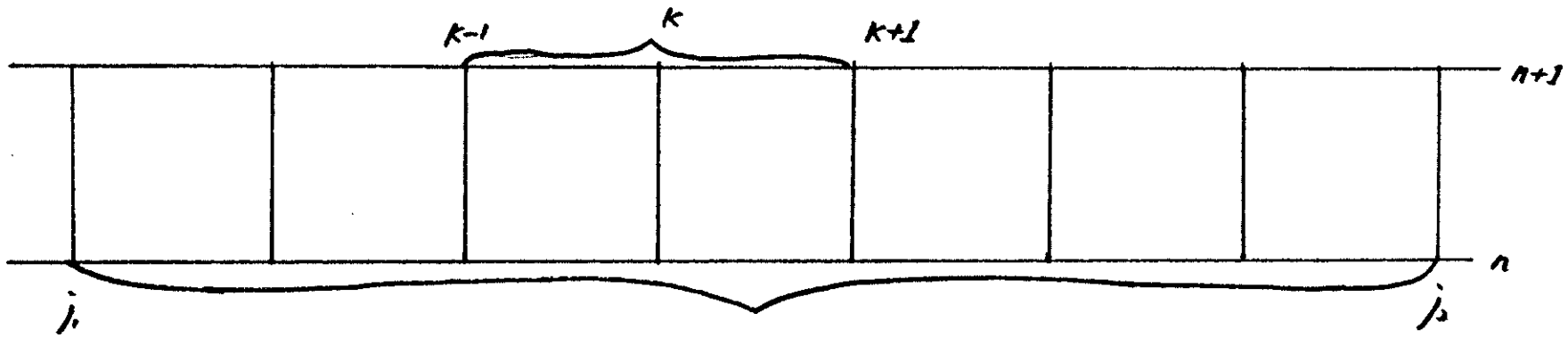


Fig. 4.2

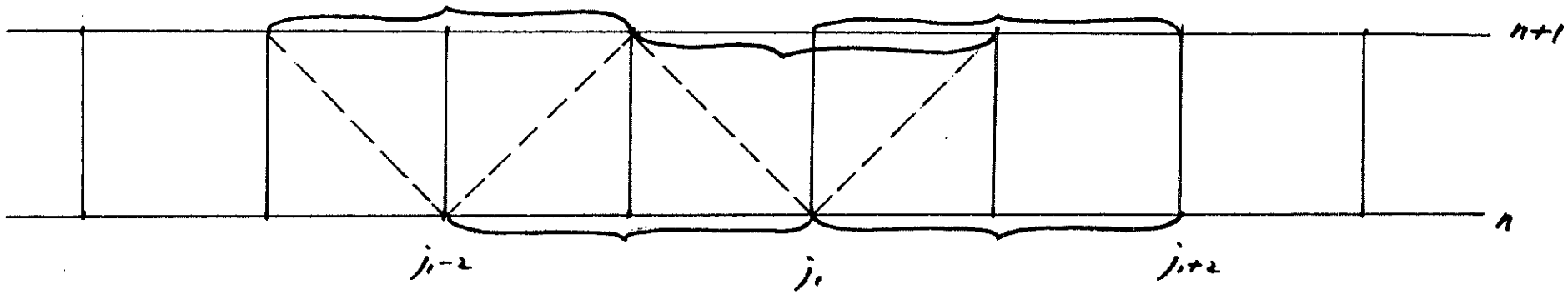


Fig. 4.3.

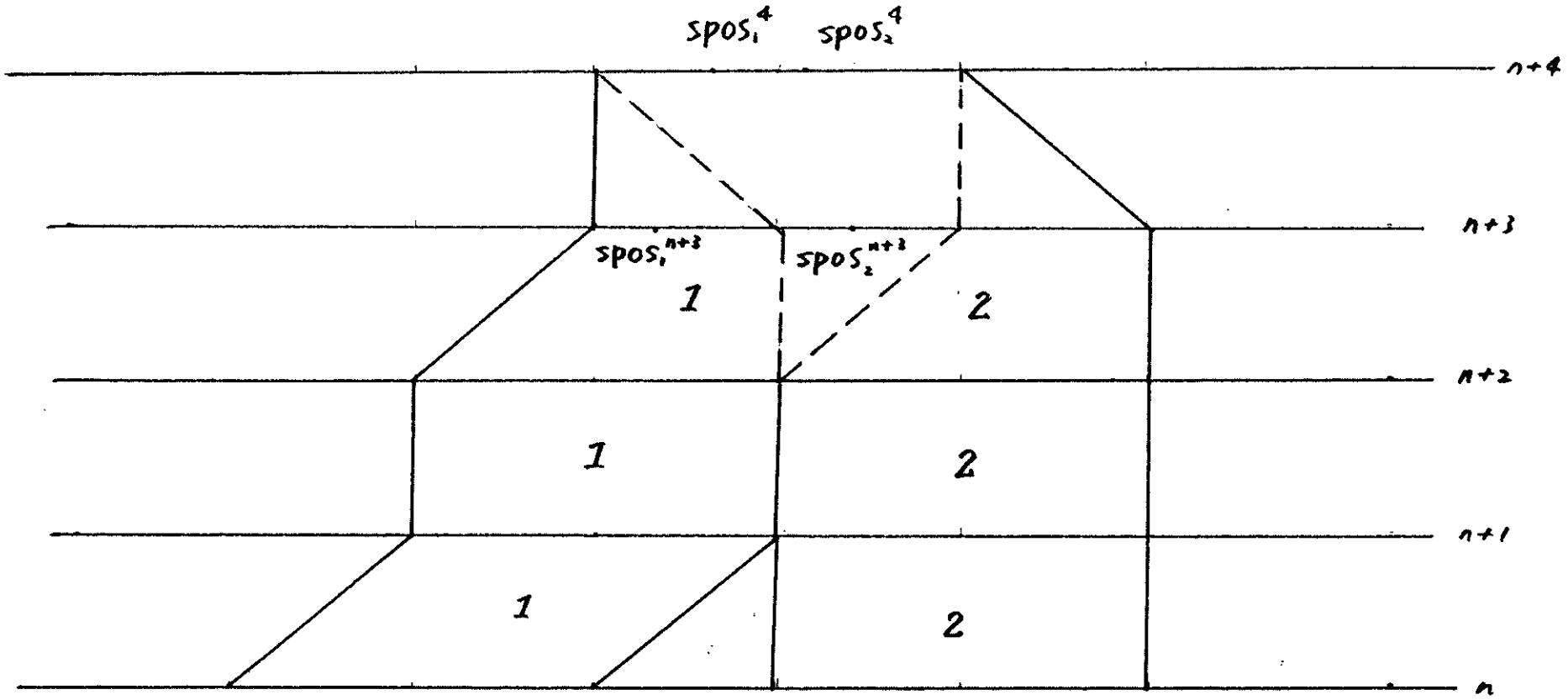
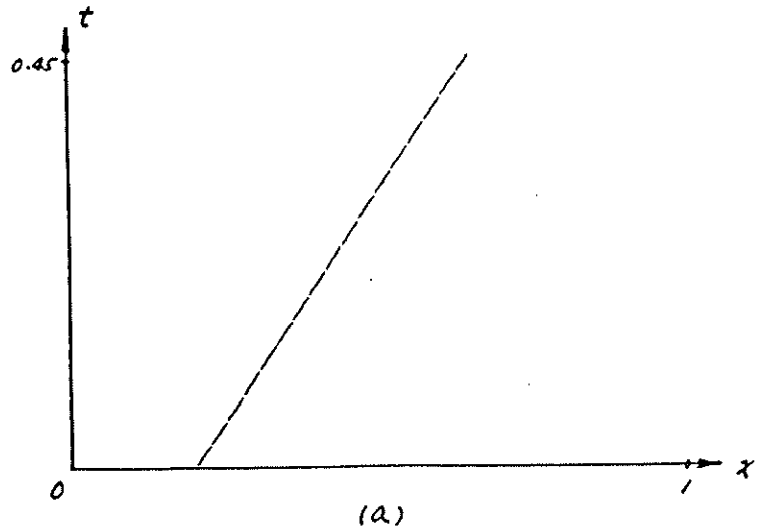
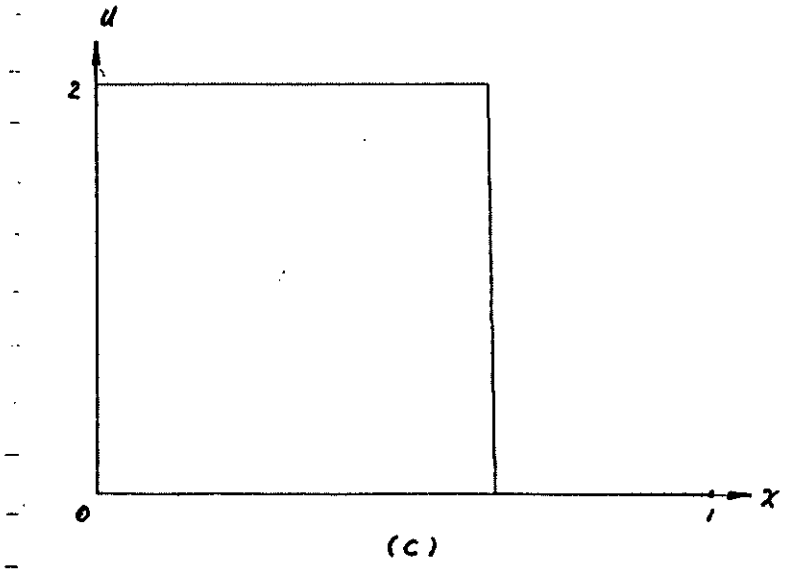


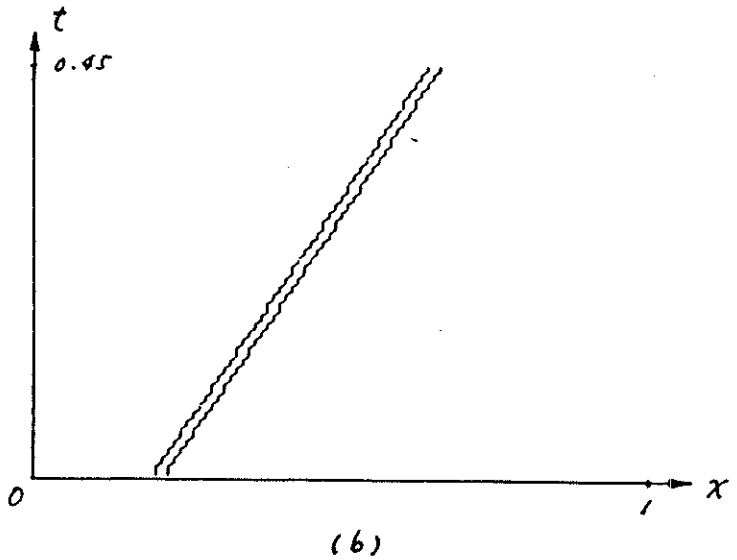
Fig. 44.



(a)



(c)



(b)

Fig. 5.1

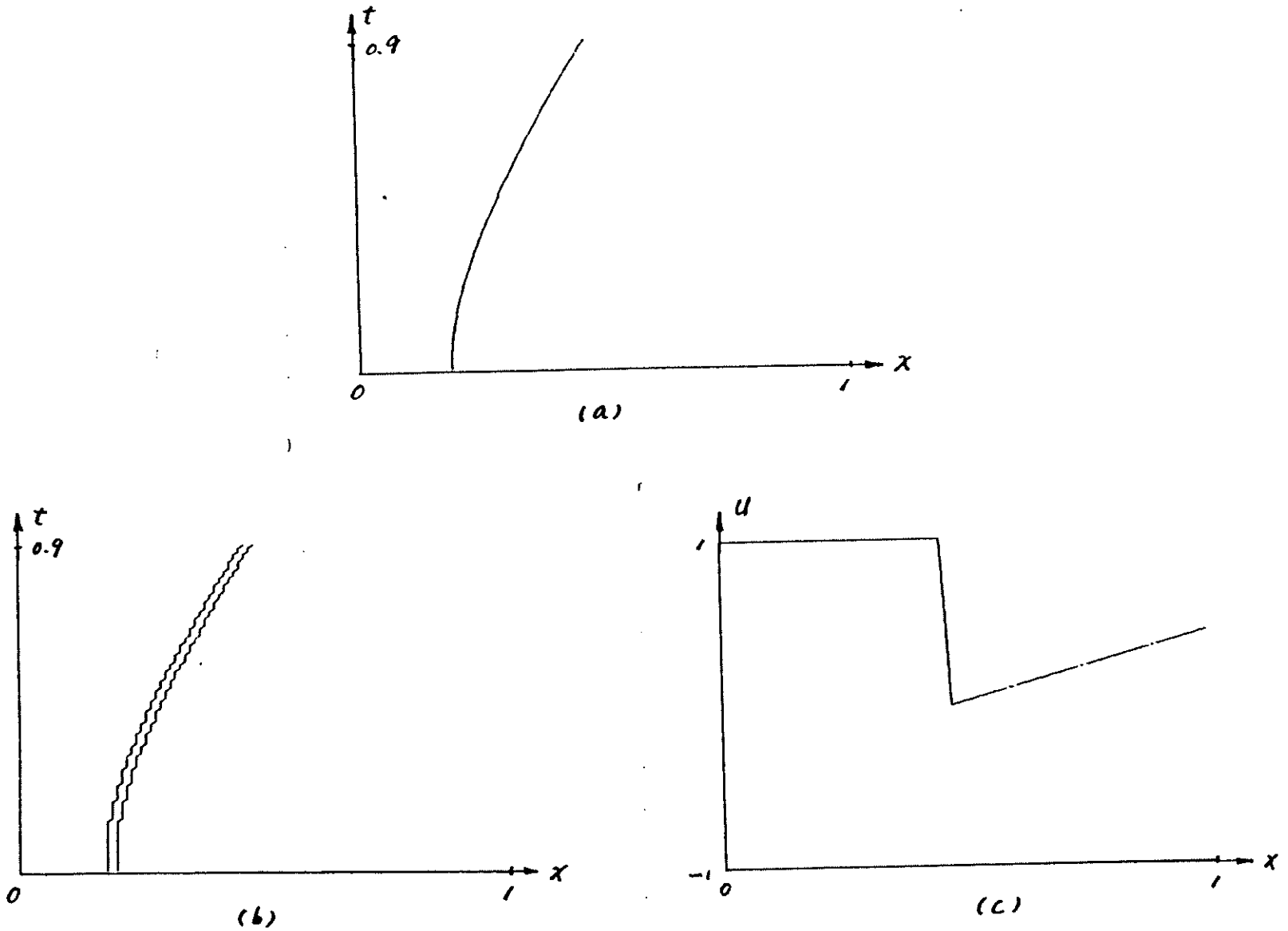
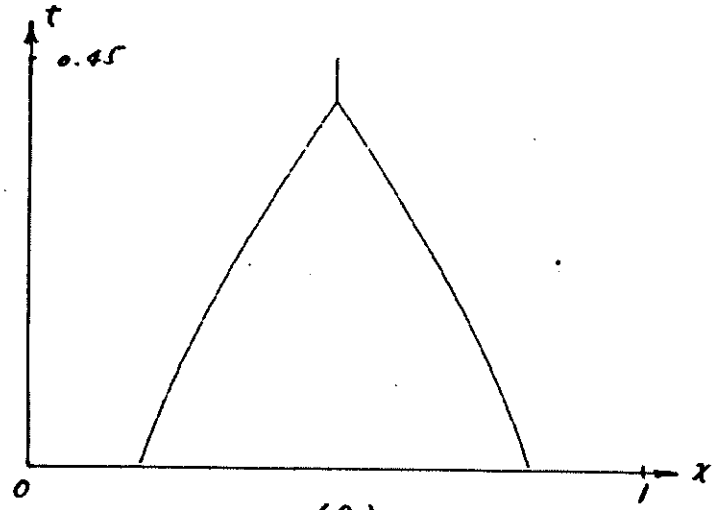


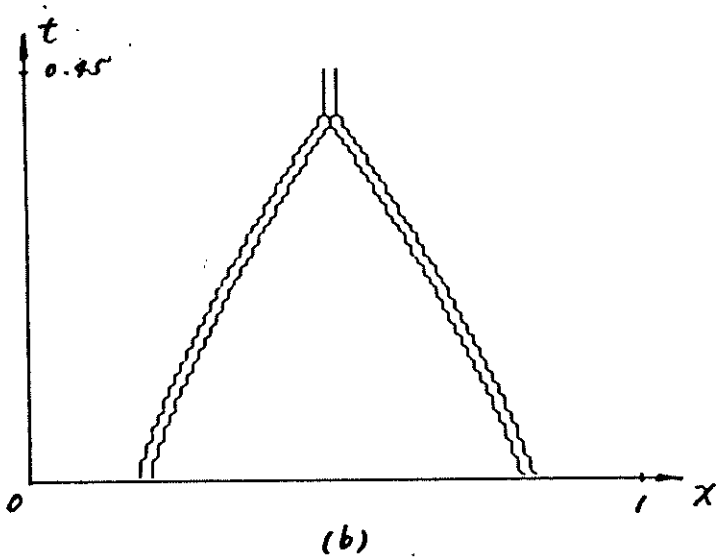
Fig. 5.2

N	AS	TS	N	AS	TS
1	0.1850000	0.1850000	51	0.2777704	0.2778890
2	0.1848590	0.1850000	52	0.2810069	0.2810741
3	0.1848456	0.1850070	53	0.2842571	0.2842931
4	0.1849597	0.1851120	54	0.2874280	0.2875459
5	0.1851972	0.1853390	55	0.2907642	0.2908321
6	0.1855531	0.1856820	56	0.2941129	0.2941499
7	0.1860224	0.1861380	57	0.2973807	0.2975011
8	0.1866005	0.1867010	58	0.3008130	0.3008840
9	0.1872829	0.1873690	59	0.3042565	0.3042970
10	0.1880656	0.1881370	60	0.3076233	0.3077409
11	0.1889448	0.1890010	61	0.3111449	0.3112140
12	0.1899165	0.1899580	62	0.3146765	0.3147180
13	0.1909773	0.1910060	63	0.3181381	0.3182500
14	0.1921238	0.1921410	64	0.3217439	0.3218111
15	0.1933530	0.1933610	65	0.3252530	0.3253999
16	0.1946616	0.1946620	66	0.3289256	0.3290170
17	0.1958960	0.1960430	67	0.3326062	0.3326610
18	0.1974163	0.1975000	68	0.3361955	0.3363319
19	0.1989854	0.1990320	69	0.3399448	0.3400301
20	0.2006120	0.2006370	70	0.3437007	0.3437540
21	0.2023001	0.2023110	71	0.3473810	0.3475030
22	0.2040512	0.2040540	72	0.3512013	0.3512781
23	0.2057088	0.2058640	73	0.3549249	0.3550771
24	0.2076533	0.2077380	74	0.3588040	0.3589020
25	0.2096337	0.2096750	75	0.3626886	0.3627499
26	0.2116572	0.2116740	76	0.3664894	0.3666229
27	0.2137282	0.2137320	77	0.3704333	0.3705190
28	0.2156941	0.2158480	78	0.3743822	0.3744389
29	0.2179372	0.2180220	79	0.3782674	0.3783811
30	0.2202104	0.2202510	80	0.3822723	0.3823460
31	0.2225187	0.2225330	81	0.3861967	0.3863341
32	0.2248660	0.2248690	82	0.3902536	0.3903430
33	0.2271263	0.2272560	83	0.3943141	0.3943739
34	0.2296263	0.2296940	84	0.3983130	0.3984270
35	0.2321517	0.2321810	85	0.4024254	0.4025011
36	0.2347066	0.2347160	86	0.4064618	0.4065961
37	0.2371746	0.2372980	87	0.4106224	0.4107120
38	0.2398612	0.2399270	88	0.4147858	0.4148470
39	0.2425707	0.2426000	89	0.4188945	0.4190041
40	0.2451666	0.2453180	90	0.4231058	0.4231799
41	0.2479926	0.2480790	91	0.4272498	0.4273760
42	0.2508387	0.2508820	92	0.4315058	0.4315911
43	0.2537078	0.2537270	93	0.4356813	0.4358260
44	0.2564766	0.2566130	94	0.4399811	0.4400791
45	0.2594624	0.2595390	95	0.4442822	0.4443510
46	0.2624654	0.2625030	96	0.4485277	0.4486420
47	0.2653600	0.2655070	97	0.4528716	0.4529511
48	0.2684621	0.2685470	98	0.4571498	0.4572780
49	0.2715803	0.2716250	99	0.4615344	0.4616229
50	0.2747160	0.2747390	100	0.4658427	0.4659860

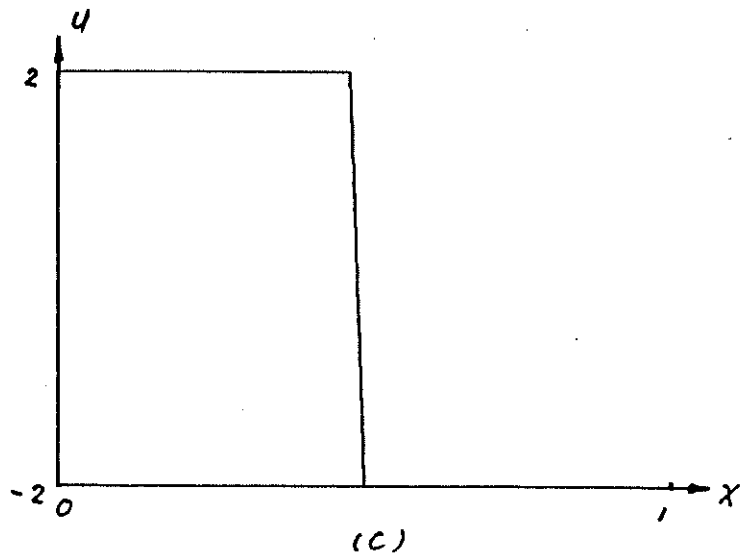
Table 5.1



(a)



(b)



(c)

Fig. 5.3

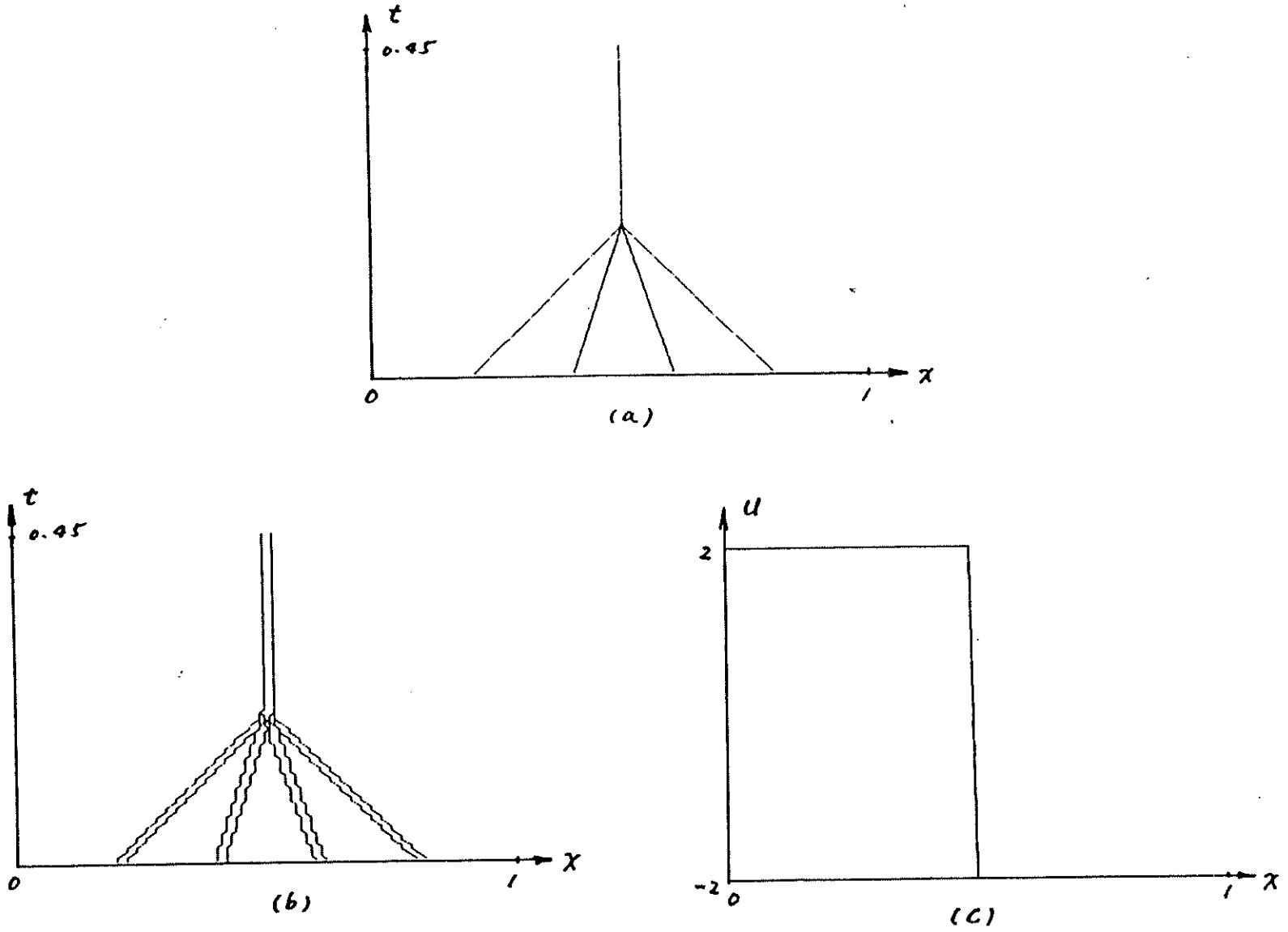
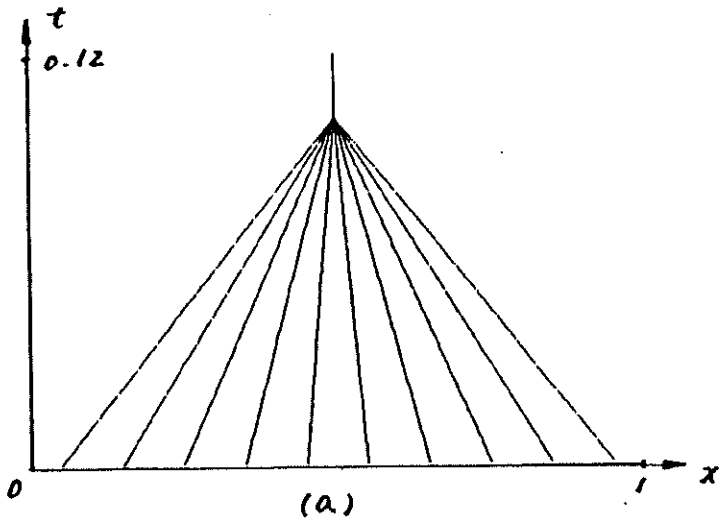
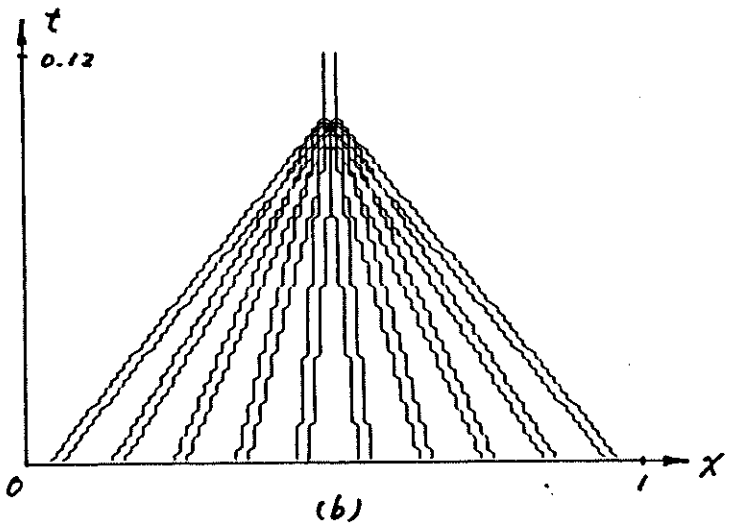


Fig. 5.4





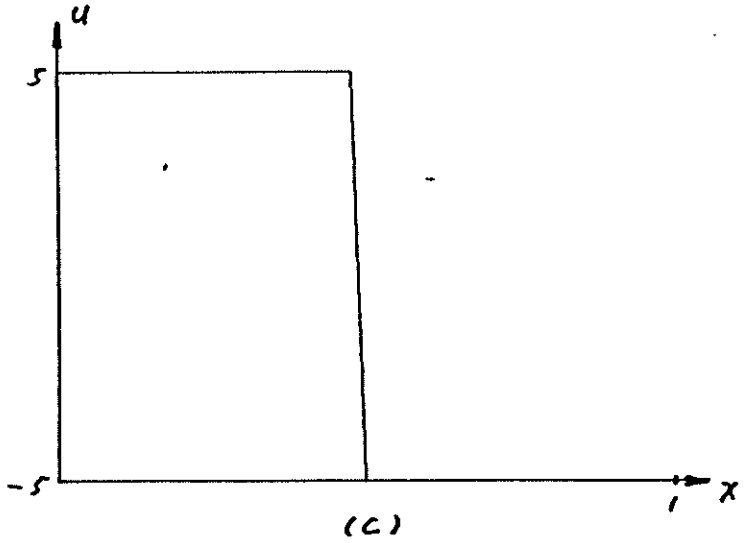
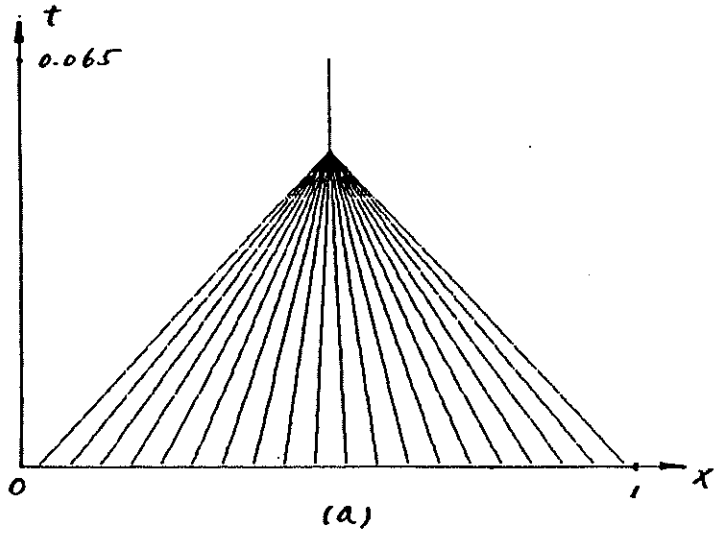
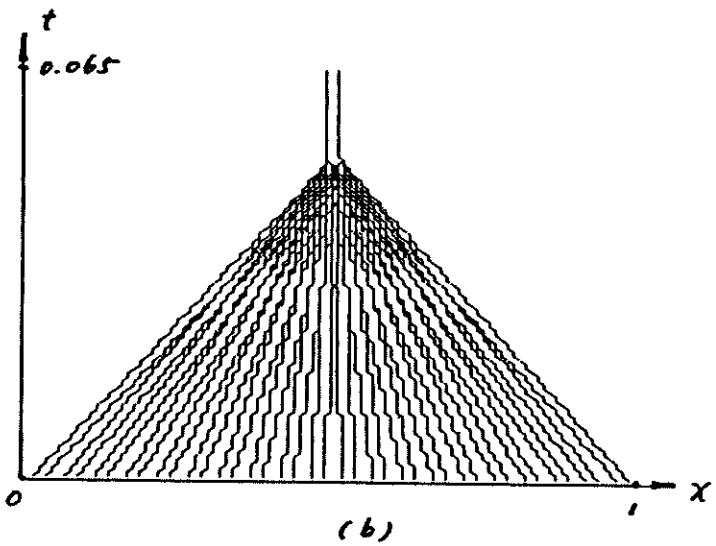


Fig. 5.5





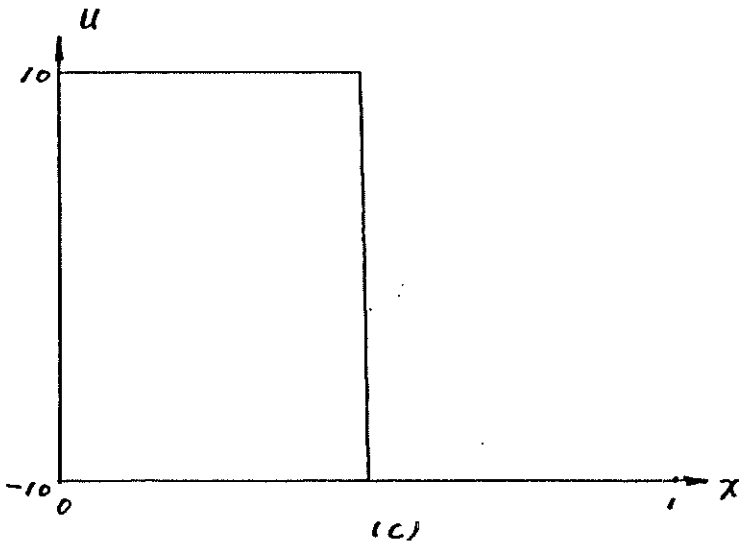


Fig. 5.6

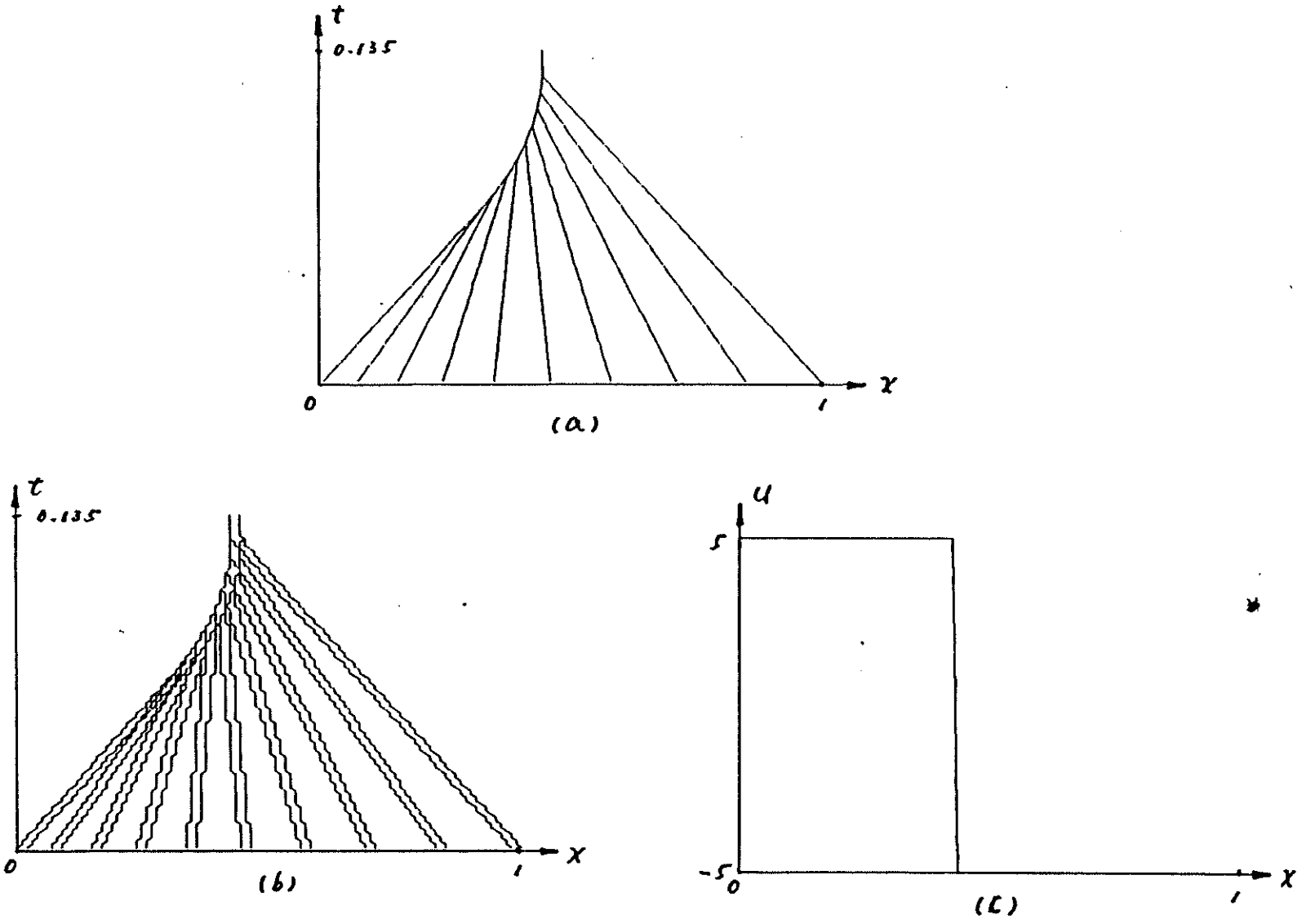
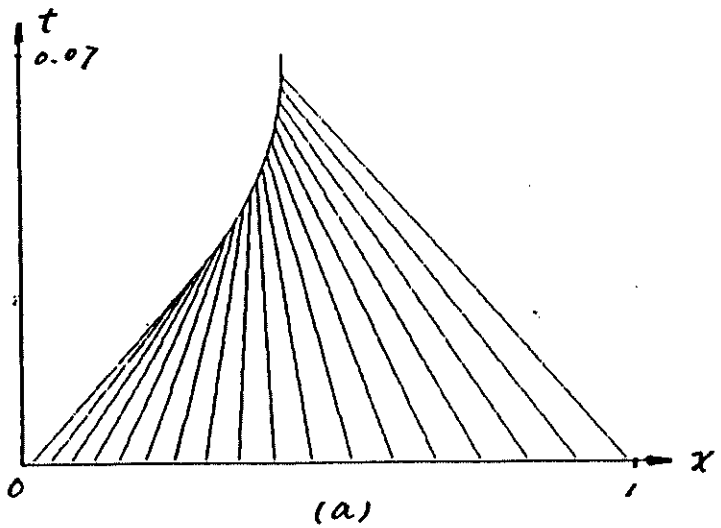
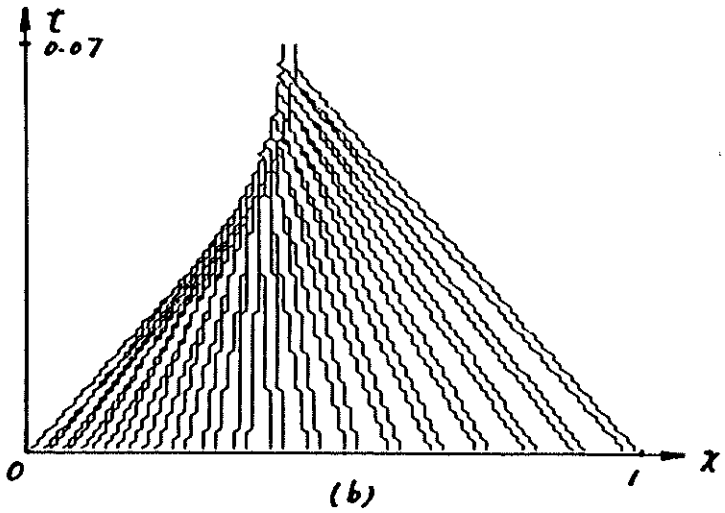


Fig. 5.7





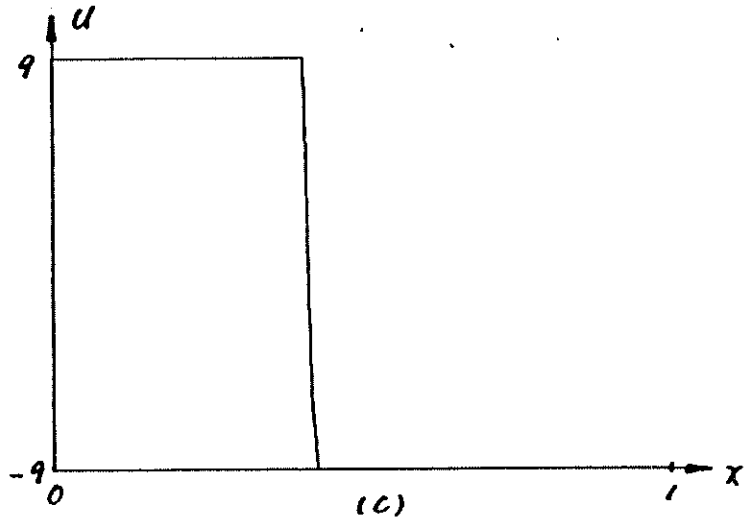


Fig. 5.8

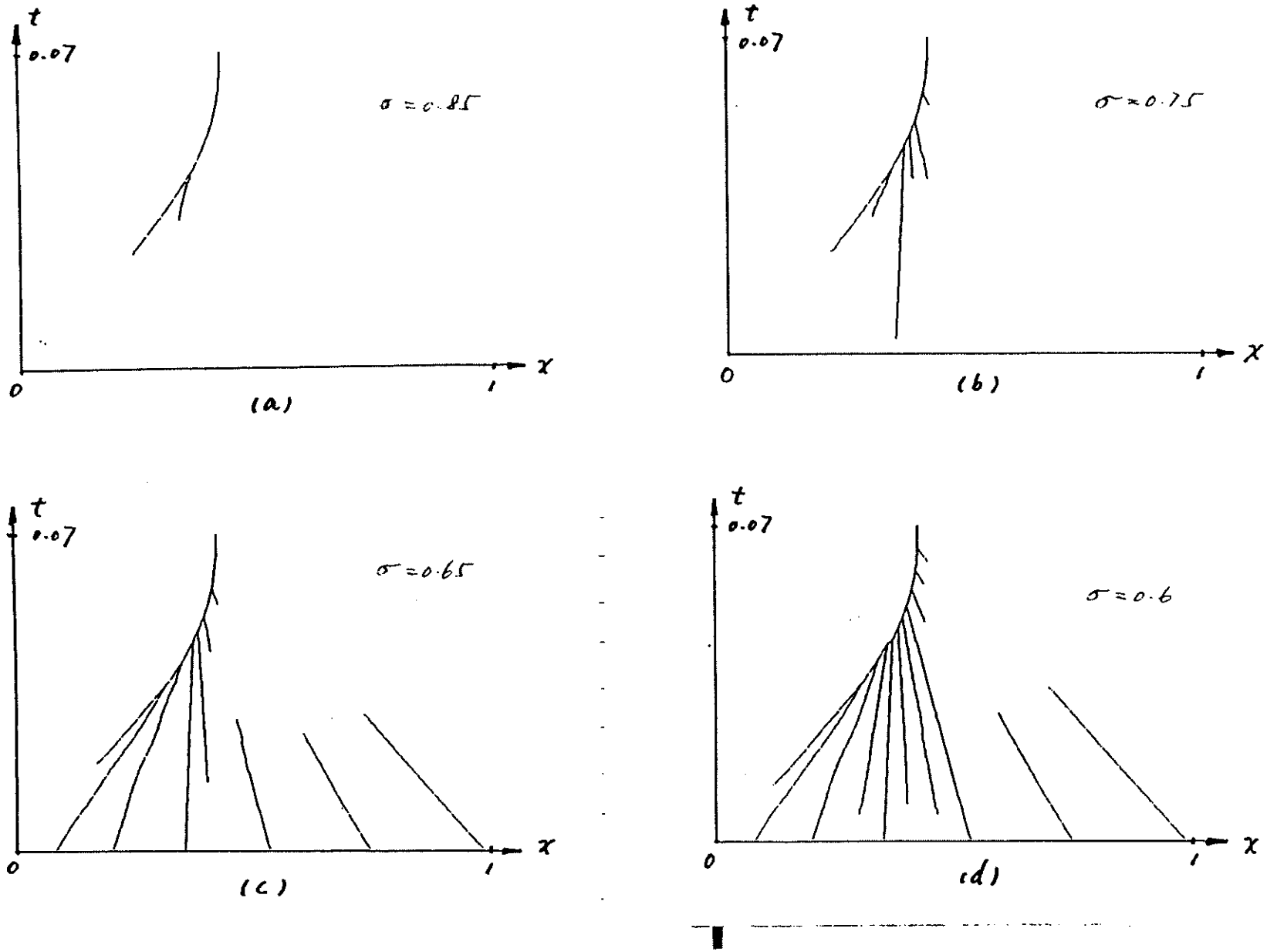
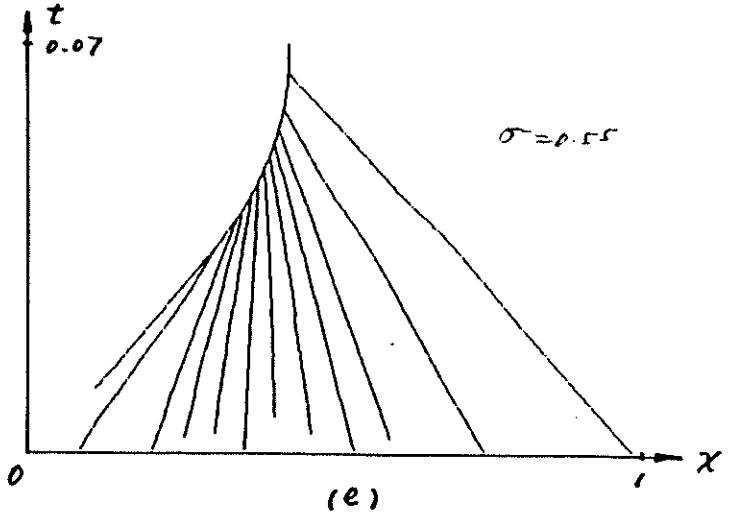


Fig. 5.9



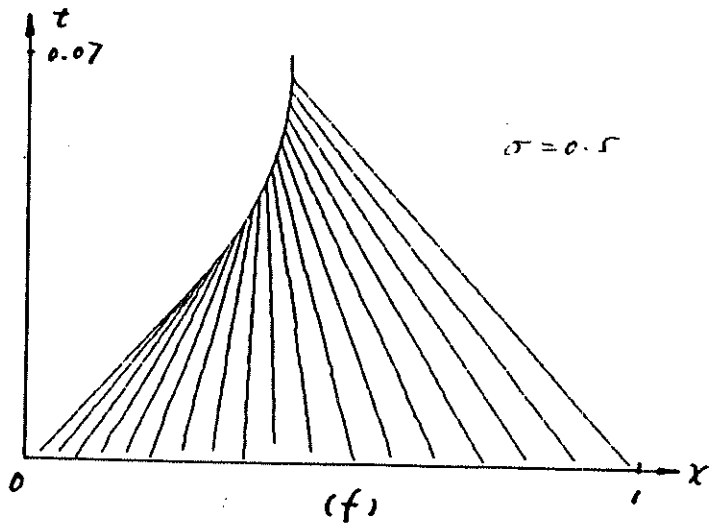


Fig. 5.9