Computing 38, 101 – 115 (1987)

Computing

On Simulation Methods for Solving the Boltzmann Equation

H. Ploss, Bregenz

Received August 24, 1986

Abstract — Zusammenfassung

On Simulation Methods for Solving the Boltzmann Equation. We will give an overview of the problem and then show how to modify an already present algorithm, so that it is also useful from a practical point of view. Further we will show that this algorithm is suitable for parallel computing and prove by means of an example that it can compete with the Bird algorithm, which is almost always applied by engineers but has only a heuristic base.

Key words: Simulation of the Boltzmann equation, Bird algorithm, Nanbu algorithm, modified Nanbu algorithm.

Über Simulationsmethoden zur Lösung der Boltzmanngleichung. Wir geben einen Überblick über die Problemstellung und zeigen dann, wie man einen bereits vorhandenen Algorithmus modifizieren kann, so daß er auch praktisch verwendbar wird. Wir zeigen weiter, daß dieser Algorithmus für Parallelrechner geeignet ist und weisen anhand eines Beispiels seine Konkurrenzfähigkeit gegenüber dem von den Ingenieuren fast ausschließlich verwendeten Bird-Algorithmus nach.

1. Introduction

A molecular approach to the study of a gas flow is necessary if the Knudsen number Kn defined by

$$\mathrm{Kn} = \frac{\lambda}{D} = \frac{\text{mean free path of a molecule}}{\text{characteristic dimension of the flow}}$$

is of order unity or higher. A high Knudsen number may result from either a large mean free path or a very small characteristic dimension. The former is usually the case. It is a consequence of very low density (rarefied gas). Re-entry flight through the upper atmosphere has focused attention on the importance of this subject. The alternative requirement of a very small characteristic dimension can be met at any density, and for example the molecular approach is required for the study of the internal structure of shock waves.

A molecular description is provided by the kinetic theory of gases. Basic to the kinetic theory is the real valued velocity distribution function $f(t,x,v)$ where $t\in\mathbb{R}$ is the time, $x\in\mathbb{R}^3$ is the position and $v\in\mathbb{R}^3$ is the velocity. If $A\subset\mathbb{R}^3$ and $B\subset\mathbb{R}^3$ then

$$\int_B \int_A f(t,x,v)\,dx\,dv$$

can be interpreted as the expected number of molecules in A with velocities in B at time t. The time evolution of $f(t,x,v)$ in the absence of external forces is described by

$$\frac{\partial f}{\partial t}+\langle v, \operatorname{grad}_x f\rangle = Jf \tag{1}$$

where

$$Jf(t,x,v)=$$

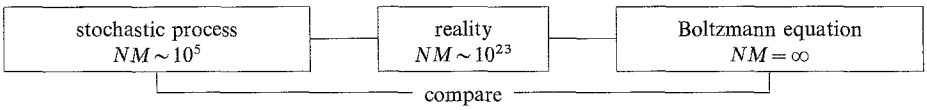
$$=\int_{\mathbb{R}^3}\int_{S^2}[f(t,x,v')f(t,x,w')-f(t,x,v)f(t,x,w)]\cdot\sigma(\eta,g)g\;d\omega(\eta)\,dw,$$

S^2 the unit sphere, $\eta\in S^2$, $g=\|v-w\|$,

$$v'=\frac{1}{2}[(v+w)+g\eta],\quad w'=\frac{1}{2}[(v+w)-g\eta].$$

The mapping $\eta\mapsto\sigma(\eta,g)$ is called the differential cross section for g. Since Ludwig Boltzmann established the non-linear integro-differential-equation (1) in 1872 generations of mathematicans and physicists have worked on it. (Even David Hilbert was concerned with this equation.)

The enormous mathematical difficulties associated with the Boltzmann equation (1) generally preclude direct approaches that would lead to exact analytical solutions. This has meant that a large number of indirect approaches have been proposed (see e.g. [1]). The disadvantages of many of the resulting methods such as uncontrollable approximations and/or enormous computational efforts are so serious that there was a great need for a search of new methods. By means of ever more powerful computers it has become possible to develop simulation procedures which imitate the behavior of a gas flow. The general idea of any simulation method is to construct a stochastic process for a NM-particle system in such a way that the NM-particle distribution at time t is "near" the solution $f(t,.,.)$ of the Boltzmann equation.



Such a procedure was first realized in the 1960s by G. A. Bird [2] and he called it "direct Monte Carlo simulation method". This Bird scheme is still the main one being used in engineering problems, although Belotserkovskiy-Yanitzkiy [3] in 1975, Deshpande [3] in 1978 and Nanbu [4] in 1980 presented alternatives with a better theoretical basis. The reason for that is that the Bird scheme provided good results in comparison with experiments and had no rival in computing times. The

computational effort is of crucial importance in the simulation and a little more complicated problems could only be calculated with the Bird scheme. We present here a modification of the Nanbu procedure which in use for parallel processors is comparable in computational efforts to the Bird simulation but still retains the level of rigour.

2. Description of the Simulation Method

Starting ideas are the following: Time is advanced by discrete steps of magnitude Δt in order to treat the flowterm $\langle v, \mathrm{grad}_x f\rangle$ and the collision term Jf separately. The position space Λ is divided into cells. The dimensions of the cells are chosen so that spatial homogeneity can be assumed in each cell.

By using equation (1) we can write

$$\int_0^{\Delta t} (f_t + \langle v, \mathrm{grad}_x f\rangle)(t, x+vt, v)\, dt = \int_0^{\Delta t} \frac{d}{dt} f(t, x+vt, v)\, dt =$$

$$= \int_0^{\Delta t} Jf(t, x+vt, v)\, dt \approx \Delta t \cdot Jf(0, x, v).$$

Then follows

$$f(\Delta t, x, v) \approx f(0, x-\Delta t \cdot v, v) + \Delta t \cdot Jf(0, x-\Delta t \cdot v, v).$$

And by defining

$$Tf(., x, v) := f(., x-\Delta t \cdot v, v)$$

we obtain

$$f(\Delta t, x, v) \approx T(1+\Delta t \cdot J) f(0, x, v).$$

We define

$$f_c(0, x, v) := \begin{cases} f(0, x, v) & x \in C \\ 0 & x \notin C \end{cases} \quad \text{for each cell } C.$$

We assume spatial homogeneity in each cell:

$$x \mapsto f_c(0, x, v) \qquad \text{is constant;}$$
$$f_c(0, v) := f_c(0, x, v) \text{ for each cell } C.$$

Thus $f(\Delta t, x, v)$ is calculated in two steps:

(a) Calculate

$$\tilde{f}_c(\Delta t, v) = (1+\Delta t \cdot J) f_c(0, v) \tag{2}$$

for each cell C. This is a time step approximation for the spatially homogeneous Boltzmann equation. It describes the effect of collisions on the velocities.

$$\tilde{f}(\Delta t, x, v) := \sum_{\text{Cells}} (1+\Delta t \cdot J) f_c(0, v)$$

(b) Substitute $x \mapsto x - \Delta t \cdot v$:

$$f(\Delta t, x, v) \approx T\tilde{f}(\Delta t, x, v) = \tilde{f}(\Delta t, x-\Delta t \cdot v, v)$$

It describes the collisionless motion of the particles over the time interval Δt.

This leads to the following simulation procedure by means of a NM-particle system.

(i) Divide the position space Λ in a suitable way into cells. Sample the positions x_i and the velocities v_i for the NM-testparticles according to $f(0, x, v)$. Choose a time step Δt, small compared to the mean free time per molecule.

(ii) Consider the cell C. There are, say, N particles in C. During the collision process we "ignore" the positions and consider only the velocities $(v_1, \ldots, v_N)$ of the N particles. I.e. we approximate $f_c(0, v)$ by

$$\frac{n}{N} \sum_{j=1}^{N} \delta(v - v_j),$$

where $n = \frac{N}{V}$ is the particle density in the cell, V the volume of the cell and δ the Dirac delta function.

Find a stochastic game transforming $(v_1, \ldots, v_N)$ into $(v_1(\Delta t), \ldots, v_N(\Delta t))$ so that

$$\frac{n}{N} \sum_{j=1}^{N} \delta(v - v_j(\Delta t))$$

is an approximation for $\tilde{f}_c(\Delta t, v)$.

Do this for all cells C.

(iii) There is no question of how to simulate step (b):

$$x_i \mapsto x_i + \Delta t \cdot v_i \quad (v_i = v_i(\Delta t);\ 1 \leq i \leq NM).$$

Compute interactions with boundaries as required.

(iv) Replace the flow time t by $t + \Delta t$ and if the flow time is smaller than a required time T, go to (ii).

Otherwise sample the flow properties.

The treatment of the collision process (ii) is of crucial importance for the whole simulation method and here the various procedures mentioned above differ.

3. The Collision Algorithm of Bird and Nanbu

We consider a cell C with N testparticles with velocities $v_1, \ldots, v_N$. We use the following notations:

$$\sigma_\tau(g) := \int_{S^2} \sigma(\eta, g)\, d\omega(\eta) \quad \text{(total cross section)},$$

$$g_{ij} := \| g_i - g_j \|,$$

$$(\sigma_\tau(g) \cdot g)_{\max} := \max \{\sigma_\tau(g_{ij}) g_{ij};\ 1 \leq i < j \leq N\},$$

$$\overline{\sigma_\tau(g) \cdot g} := \frac{1}{N^2} \sum_{i,j=1}^{N} \sigma_\tau(g_{ij}) g_{ij}.$$

Using (l) and (l,m) we denote the particle with velocity (v_l) and the particle pair with velocities (v_l, v_m) respectively.

3.1 The Bird-Algorithm

Bird devised his algorithm through reference to the physics of molecular collisions and he made a thorough study on the reduction of computing time. We note that from a practical point of view it is the most successful method, used in many applications ([2], [5]). But from a rigorous point of view it is only heuristic. For a more detailed discussion we refer to [3] and our report [6].

(i) To sample a collision pair (l,m) Bird employed the acceptance-rejection method: Generate a random fraction R_f and accept a randomly chosen pair (l,m) as a collision pair if

$$\frac{\sigma_\tau(g_{lm})\, g_{lm}}{g^*} \geq R_f$$

where $g^* \geq (\sigma_\tau(g)\, g)_{\max}$.

Otherwise repeat the selection.

(ii) Let (l,m) be an accepted pair. Advance the time by

$$\tau_1 = \frac{2}{N n \cdot \sigma_\tau(g_{lm})\, g_{lm}}.$$

[Remark: $\frac{1}{n \cdot \sigma_\tau(g)\, g}$ is the mean collision time per molecule and $\frac{2}{N n \cdot \sigma_\tau(g)\, g}$ can be interpreted as the mean free time between two collisions.]

(iii) Replace the two molecular velocities of the pair (l,m) by the post collision velocities:

$$v_l \mapsto v_l' = \frac{1}{2}\left[(v_l + v_m) - \eta \cdot g_{lm}\right]$$

$$v_m \mapsto v_m' = \frac{1}{2}\left[(v_l + v_m) + \eta \cdot g_{lm}\right]$$

where $\eta \in S^2$ is chosen randomly in correspondence with the collision law (e.g. from a uniform sphere distribution in the case of a hard sphere gas). The velocities of other molecules are unchanged and hence all molecular velocities at time $t + \tau_1$ are known.

(iv) Repeat steps (i) to (iii) until the condition

$$\tau_1 + \tau_2 + \ldots + \tau_{N_s} \geq \Delta t \tag{3}$$

is satisfied. Condition (3) determines the number of collisions N_s in $[t, t+\Delta t]$.

3.2 The Nanbu-Algorithm

Nanbu obtained his algorithm by inserting

$$f_c(0,v)=\frac{n}{N}\sum_{j=1}^{N}\delta(v-v_j)$$

in equation (2) and a formal calculation [4]. He ended up with the following procedure.

(i) For the i-th particle with velocity v_i compute

$$P_i:=\Delta t\frac{n}{N}\sum_{j=1}^{N}\sigma_\tau(g_{ij})g_{ij}.$$

Generate a random fraction $R_f\in[0,1]$ and

$$\text{if}\begin{cases}R_f>P_i \text{ (rejected): } v_i'=v_i \text{ (no collision),}\\ R_f<P_i \text{ (accepted) go to, in turn, steps (ii) and (iii).}\end{cases}$$

(ii) Sample a collision partner (j) from the conditional probability distribution

$$\{P_{ik}^*;\ k=1,2,\ldots,i-1,i+1,\ldots,N\}$$

where

$$P_{ik}^*:=\frac{P_{ik}}{P_i},\quad P_{ik}=\Delta t\frac{n}{N}\sigma_\tau(g_{ik})g_{ik}.$$

The following condition determines (j):

$$\sum_{k=1}^{j-1}P_{ik}^*<R_f'\le\sum_{k=1}^{j}P_{ik}^*$$

where R_f' is another random fraction.

(iii) After the collision partner (j) of Particle (i) is known compute

$$v_i'=\frac{1}{2}[(v_i+v_j)+\eta g_{ij}]$$

where $\eta\in S^2$ is chosen in correspondence with the collision law.

Repeat this procedure for all N particles in the cell. The realization of the Nanbu collision process is essentially different from that of Bird. The collision probability of a molecule is determined without specifying its collision partner. A particle that does not collide can appear as a "dummy collision partner". The consequence is that in Nanbu's scheme energy and momentum are only conserved in the mean, but the correlations of the velocities are reduced. And whereas in Bird's scheme the postcollision velocities immediately go to work, in Nanbu's scheme they are stored until the whole collision process in the cell is completed. However the expectation number of collisions is the same in both strategies [3].

3.3 Babovsky's Interpretation

In [7] Babovsky investigates the Nanbu algorithm and presents rigorous results concerning questions of justification. We bring here a short outline.

To approximate the solution of the spatial homogeneous Boltzmann-equation

$$\frac{\partial}{\partial t} f(t,v) = \int_{\mathbb{R}^3} \int_{S^2} \sigma(\eta, g)\, g\, [f(t,v')\, f(t,w') - f(t,v)\, f(t,w)]\, d\omega(\eta)\, dw$$

given $f(0,v) = f_0(v)$ at time $t = \Delta t$ we consider the following iteration process. We divide Δt in L intervals of magnitude $\Delta\tau$: $\Delta t = L \cdot \Delta\tau$ and solve the equation

$$\begin{cases} \dfrac{\partial}{\partial t} \tilde{f}(t,v) = \displaystyle\int_{\mathbb{R}^3} \int_{S^2} \sigma \cdot g\, [\tilde{f}(t,v')\, f(0,w') - \tilde{f}(t,v)\, f(0,w)]\, d\omega(\eta)\, dw \\ \tilde{f}(0,v) = f(0,v) = f_0(v) \end{cases} \tag{4}$$

in $[0, \Delta\tau]$. Then we replace $f(0,w)$ by $\tilde{f}(\Delta\tau, w)$ and solve (4) in $[\Delta\tau, 2\Delta\tau]$; and so on.

Hence in connection with the simulation we keep the following diagram in mind.

$$\begin{array}{ccccccccc} f(0,v) & \xrightarrow[t\in[0,\Delta\tau]]{(4)} & \tilde{f}(\Delta\tau, v) & \xrightarrow[t\in[\Delta\tau, 2\Delta\tau]]{(4)} & \cdots & \rightarrow & \tilde{f}(\Delta t, v) & \xrightarrow[L\to\infty]{\Delta\tau\to 0} & f(\Delta t, v) \\ \downarrow & & & & & & \uparrow & & \\ \dfrac{1}{V}\displaystyle\sum_{j=1}^{N} \delta(v - v_j(0)) & & & & & & \dfrac{1}{V}\displaystyle\sum_{j=1}^{N} \delta(v - v_j(\Delta t)) & & \\ \downarrow & & & & & & \uparrow & & \\ \{v_1, \ldots, v_N\}_{t=0} & \rightarrow & \{v_1, \ldots, v_N\}_{t=\Delta\tau} & \rightarrow & \cdots & \rightarrow & \{v_1, \ldots, v_N\}_{t=\Delta t} & & \end{array} \tag{5}$$

Equation (4) is a linear transport equation and describes the time evolution of a distribution function for testparticles which are distributed at time $t = 0$ according to $f(0,v)$ and hit "hosparticles", which are distributed during Δt according to $f(0,v)$. Babovsky used the fact that such a transport equation is associated with a Markov process. He showed that under certain assumptions the use of Nanbu's algorithm for

$$\{v_1, \ldots, v_N\}_{t=k\Delta\tau} \mapsto \{v_1, \ldots, v_N\}_{t=(k+1)\Delta\tau}$$

is a realization of that Markov process. But we note that at time there is no rigorous proof for the convergence of diagram (5).

3.4 The Problem of the N^2-Effort

A very serious disadvantage of the Nanbu algorithm is of practical nature. We have to compute

$$P_i = \Delta t\, \frac{n}{N} \sum_{j=1}^{N} \sigma_\tau(g_{ij})\, g_{ij} \quad \text{for } i = 1, \ldots, N$$

i.e. all relative velocities g_{ij}.

This $O(N^2)$ effort renders the procedure inefficient from a practical point of view. In fact, in a recent paper [8] Nanbu himself used the Bird algorithm. We show in 4. that by the use of a simple but effective trick we can reduce the effort to order $O(N)$.

The $O(N^2)$ effort seems to appear implicitly in the Bird algorithm as well. For we have to calculate $\binom{N}{2}$ terms $\sigma_\tau(g_{ij})\,g_{ij}$ to determine $(\sigma_\tau(g)\,g)_{\max}$ and $(\sigma_\tau(g)\,g)_{\max}$ can even undergo a change with each collision. Bird avoids the $O(N^2)$ effort by estimating g^* and if $\sigma_\tau(g)\,g > g^*$ should appear, while the program is being run, it replaces g^* and so on. This is of course a source of error with respect to the selection of a collision pair. However the practicans argue [5] that in any simulation problem a suitable g^* can readily be found during a period of adjustment.

4. The Modified Nanbu-Algorithm

As mentioned above the Nanbu algorithm in the way shown in 3.2 is inefficient from a practical point of view. We modify the algorithm in two points. The decisive improvement is in the second point and was suggested by Babovsky.

(a) As in 3.3 we divide the decoupling interval Δt in L intervals of magnitude $\Delta\tau$ and run the collision algorithm L times before going over to the collisionless motion. (Therefore in the following $P_{ij} = \Delta\tau \cdot \frac{n}{N} \cdot \sigma_\tau(g_{ij})\,g_{ij}$.) This leads to a greater flexibility in the choice of Δt and can reduce the computation time. Then during the collision process a particle can collide several times as in the Bird scheme.

(b) Obviously we can combine steps (i) and (ii) of the Nanbu-algorithm: We generate a random fraction $R_f \in [0, 1]$. If $R_f \in [P_i, 1]$ the i-th particle does not collide. If $R_f \in P_{ij}$ the i-th particle collides and the collision partner is the j-th particle (see Fig. 1).

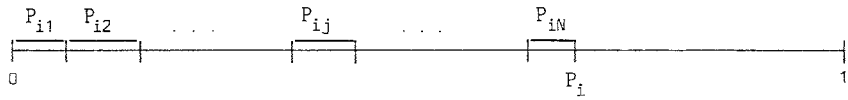


Fig. 1

It is of course irrelevant how we distribute the probabilities P_{ij} over the unit interval. We distribute them according to Fig. 2.

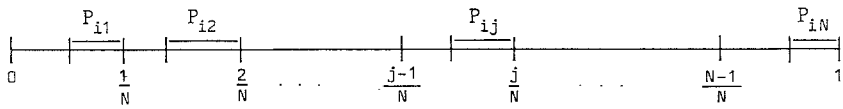


Fig. 2

I.e. we distribute $[P_i, 1]$ ($\hat{=}$ probability that the i-th particle does not collide) in such a way on the P_{ij} that N identical intervals are obtained. In this case we can immediately decide in which of the N intervals our random fraction R_f is located.

If $R_f \in \left[\frac{j-1}{N}, \frac{j}{N}\right]$ we compute only P_{ij} and

if $\begin{cases} R_f \leq \left[\frac{j}{N} - P_{ij}\right] \mapsto \text{the } i\text{-th particle does not collide,} \\ R_f > \left[\frac{j}{N} - P_{ij}\right] \mapsto \text{the } i\text{-th particle does collide and its collision partner is the } j\text{-th particle.} \end{cases}$

The Numerical Effort is Now Only of Order $O(N)$

One point must be considered. The above procedure works if we have

$$P_{ij} = \Delta\tau \cdot \frac{n}{N} \cdot \sigma_\tau(g_{ij})\, g_{ij} \leq \frac{1}{N}$$
$$\text{or } \Delta\tau \cdot n \cdot \sigma_\tau(g_{ij})\, g_{ij} \leq 1 \text{ for } 1 \leq i < j \leq N. \tag{6}$$

We have to satisfy this condition by means of a suitable choice of $\Delta\tau$ (i.e. of the decoupling interval Δt and of the parameter L), naturally without computing all $\sigma_\tau(g_{ij})\, g_{ij}$ but by means of estimations of linear effort. For example we have for the molecule models which are used in the case of a monoatomic gas (see [5])

$$\sigma_\tau(g_{ij})\, g_{ij} = c\, {g_{ij}}^\beta \text{ with } 0 < \beta \leq 1 \text{ and } c \text{ as a constant.}$$

And we can estimate ${g_{ij}}^\beta$ e.g. by

$$\begin{aligned} {g_{ij}}^\beta &= \| v_i - v_j \|^\beta = \|(v_i - m) - (v_j - m)\|^\beta \\ &\leq (\| v_i - m \| + \| v_j - m \|)^\beta \leq (\| v_i - m \| + \max_{1 \leq j \leq N} \| v_j - m \|)^\beta \end{aligned}$$

with

$$m = \frac{1}{N} \sum_{j=1}^{N} v_j.$$

In doing so we might obtain an unnecessarily small $\Delta\tau$. Therefore we suggest for practical purposes the following alternative. During a testphase $\Delta\tau$ is determined in such a way that

$$P_{ij} \leq k \frac{1}{N} \quad \text{with } k < 1 \text{ chosen}$$

is true for all actually computed P_{ij}. An error estimation has not yet been made in this connection.

In general we should choose

$$\Delta t < \frac{1}{n \cdot \sigma_\tau(g)\, g} \quad (\hat{=} \text{mean collision time per mulecule}). \tag{7}$$

With $L \sim 10$ we have

$$\Delta\tau < \frac{1}{n \cdot \overline{\sigma_\tau(g)\, g}} \quad \text{or} \quad \Delta\tau \cdot n \cdot \overline{\sigma_\tau(g)\, g} < 1$$

and our condition (6) is a specification of the general condition (7).

We designate the simulation scheme with the above described modification as N^*-scheme.

5. On the Vectorizing of the Algorithm

By "vectorizing" we mean in this context the adaption of computerprograms to the special architecture of the computer. The vector processor CYBER 205 is constructed according to the pipeline principle. In this concept high performance is achieved if the algorithms are parallel in such a way that a lot of the same operations can be performed with independent data stored in sequence.

The most time consuming part of the simulation program is the continually repeating collision process. The Bird algorithm is here recursive in the sense that after a collision the computation goes on using the postcollision velocities. We could not find a reasonable vectorizing on the CYBER 205. Remark: According to [9] the vectorizing of the Bird algorithm on the CRAY-1S is of little advantage. Additionally the conditions (e.g. favorable vector lengths) from which we can expect an effective vectorizing on the CYBER 205 vary from those on the CRAY-1S.

On the contrary the N^*-algorithm is parallel in its structure. The set of the velocities at time $t=0$ determines the set of the velocities at time $t=\Delta\tau$.

$$\{v_1, \ldots, v_{NM}\}_{t=0} \mapsto \{v_1, \ldots, v_{NM}\}_{t=\Delta\tau}$$

We will now show how we vectorized an essential part of the collision process.

To determine the collision probability of the particle (i) – we call i the number of the particle (i) – we need the possible collision partner (j) and have to compute $g_{ij} = \| v_i - v_j \|$. We want to do this "simultaneously" for all NM particles which are in the computing field. In doing so we must keep in mind that only a particle (j) from the same cell comes in question as a possible collision partner for particle (i). Let $NCELLS$ be the number of cells which are numbered in sequence and L_c be the cell-number in which the particle numbered L is found.

The x, y, z-velocity-components of the particles are stored in the arrays VX, VY, VZ. Our goal is to create arrays VXG, VYG, VZG which correspondingly contain the velocity-components of the possible collision partners. After that we can immediately vectorize

$$SQRT((VX(I) - VXG(I))**2 + (VY(I) - VYG(I))**2 + \\ + (VZ(I) - VZG(I))**2) \quad (1 \leq I \leq NM).$$

As preparation the arrays R, NN, NB, $LIST$ were created. NM random fractions are stored in R.

$$NN(I) = \text{number of particles in cell } I \ (1 \leq I \leq NCELLS)$$

$$NB(1) = 0,\ NB(J) = NN(1) + \ldots + NN(J-1) \ (2 \leq J \leq NCELLS)$$

The number of the particles are arranged in $LIST$ according to the cells in which the particles are located and within each cell according to size: No L is stored in front of No K if

$$L_c < K_c \text{ and in case } L_c = K_c \text{ if } L < K.$$

If $LIST(J) = M$ – the particle number M is located at J in $LIST$ – we call J the index of the particle numbered M.

$$N1(NB(I)+J) = NN(I) \quad (1 \leq I \leq NCELLS\ (1 \leq J \leq NN(I)))$$
(the inner loop is vectorizable)

$$N2(J) = R(J) * N1(J) + 1 \quad (1 \leq J \leq NM)$$
(immediately vectorizable)

$$N3(NB(I)+J) = N2(NB(I)+J) + NB(I) \quad (1 \leq I \leq NCELLS\,(1 \leq J \leq NN(I)))$$
(the inner loop is vectorizable)

$N3(J)$ is the index of the possible collision partner of the particle indexed J.

$$N4(J) = LIST(N3(J)) \quad (1 \leq J \leq NM)$$
(vectorizable by the vector intrinsic function $GATHR$)

$N4(J)$ is the number of the possible collision partner of the particle indexed J.

$$N5(LIST(J)) = N4(J) \quad (1 \leq J \leq NM)$$
(vectorizable by the vector intrinsic function $SCATR$)

The number of the possible collision partner is stored according to the number of the particle. And finally

$$\left.\begin{aligned} VXG(J) &= VX(N5(J)) \\ VYG(J) &= VY(N5(J)) \\ VZG(J) &= VZ(N5(J)) \end{aligned}\right\} \quad \begin{aligned} &(1 \leq J \leq NM) \\ &\text{(vectorizable by the vector} \\ &\text{intrinsic function } GATHR) \end{aligned}$$

Through the vectorization the whole simulation program for the shock wave problem (see 6.) – using the N^*-algorithm – became faster by factor 7 than a scalar version of the program on the CYBER 205.

6. Comparison Data

1. In testcalculations in which only the collision process was considered we obtained (with nearly the same results [6]) the following computation efforts:

On the universal computer at Kaiserslautern University (Siemens S 7551, Fortran 77, 24 bit mantissa):

Table 1

Particles	Nanbu	N^*	Bird	
100	56.7	3.1	2.3	
1000	5422	17.5	8.9	CPU sec
10000	?	180	79	

On the vector processor CYBER 205 at Karlsruhe University (Vector Fortran 48 bit mantissa):

Table 2

Particles	N^*	Bird	
100	0.035	0.031	
1000	0.145	0.120	CPU sec
2000	0.263	0.214	

Remark: At the time the calculations were done we had only little experience with the CYBER 205 and we are sure that it is possible to improve the "N^*-times".

II. As a serious testexample we used the Bird and N^*-simulation scheme to compute a stationary plane shock wave for various Mach numbers in the case of a monoatomic gas. Also Nanbu calculated the shock wave problem [10]. But due to the computational effort he had to restrict himself to Maxwell-molecules. In this special case $\sigma_\tau(g_{ij})\, g_{ij}$ is independent of (i,j) and the computational effort is of order $O(N)$ even when using the original Nanbu algorithm. But from a physical point of view Maxwell-molecules are unrealistic and it is senseless to compare Nanbu's results with experiments. The Bird simulation and the N^*-simulation are of order $O(N)$ for any molecule model. For our calculations we used the VHS model (variable diameter hard sphere). Here the collision probabilities were calculated by means of a cut-off $r^{-\alpha}$ potential, but the actual collision behavior was taken from the hard sphere model. For a discussion of the VHS model we refer to [5]. Our results are compared with measurements made by H. Alsmeyer in 1975 [11].

For a plane shock wave we have gradients only in one direction and so the problem is (spatial) "1-dimensional". In a frame of reference in which the shock wave is stationary we have the following picture.

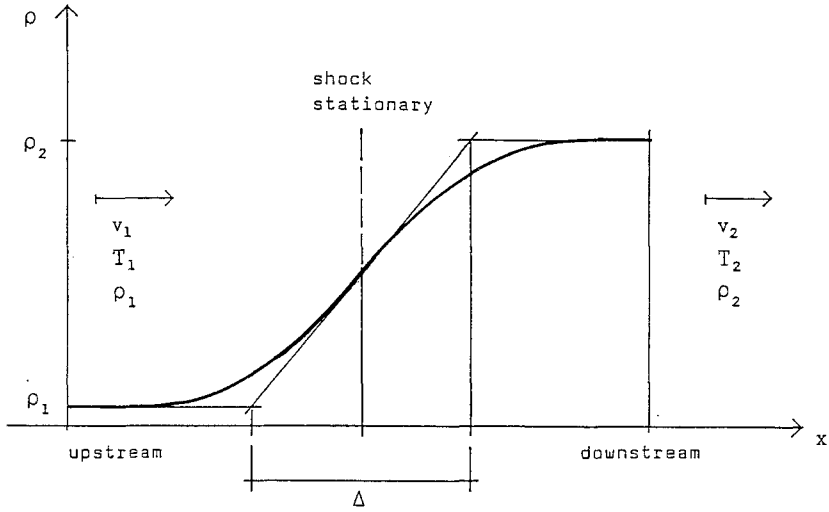


Fig. 3

The characteristic dimension of the problem is the shock width Δ defined by

$$\Delta := \frac{\rho_2 - \rho_1}{\left(\frac{d\rho}{dx}\right)_{\max}}.$$

The upstream and downstream velocities, temperatures and densities v_1, T_1, ρ_1, and v_2, T_2, ρ_2 respectively are related by the Rankine Hugoniot conditions (see e.g. [1]). A shock Mach number M_s is defined by

$$M_s := \frac{v_1}{c_1}$$

where c_1 is the speed of sound in the upstream gas.

For a description of the simulation procedure, a discussion of the simulation parameters and more results we refer to our report [6]. But the following choice of parameters, the result shown in Fig. 4 and the computational effort shown in Table 3 are typical.

40000 particles were used in the simulation; in dimensionless variables calculations were done for $-10 \leq x \leq 10$, with a cell size $\Delta x_c = 0.2$. The time step was chosen as $\Delta t = 0.1$ and the parameter L for the N^* simulation as $L = 10$. The calculations were done up to time $t = 24$. Time averages were taken after 60 runs.

Computations were done on the CYBER 205 vector processor at Karlsruhe University. The system of billing used there is SBU (System Billing Unit).

Table 3

$M_s = 3.8$	CPU sec	SBU
Bird	12900	50000
N^*	11700	54000

As seen in Table 3 the relation of CPU sec to SBU is more favorable in the predominatly scalar calculated Bird scheme than in the mainly vectorized N^*-scheme. The reason is that a high degree of vectorizing often requires more storage space.

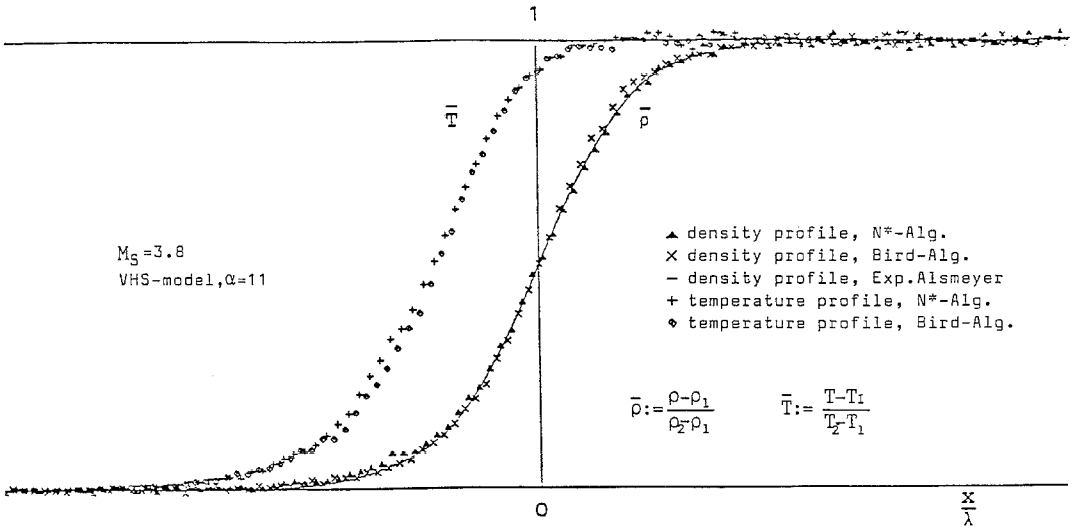


Fig. 4

7. Conclusion

Up to the present a comparison of the different simulation schemes of the Boltzmann equation could not be made in almost all cases due to the excessive computational effort. The Bird-algorithm, which is only heuristic based, had no rival in application. We hope to have shown that the N^*-algorithm is a process which is useful from a practical point of view and additionally has a good chance of becoming theoretically based.

Acknowledgement

The author wishes to thank Prof. H. Neunzert who supervised the project and his colleagues H. Babovsky and T. Mietzner (all from Kaiserslautern University) for their support.

References

[1] Cercignani, C.: Theory and Application of the Boltzmann Equation. New York: Elsevier 1975.
[2] Bird, G. A.: Molecular Gas Dynamics. Oxford: Clarendon Press 1976.
[3] Nanbu, K.: Interrelations between various direct simulation methods for solving the Boltzmann equation. J. Phys. Soc. Japan *52*, 3382 (1983).
[4] Nanbu, K.: Direct simulation scheme derived from the Boltzmann equation. J. Phys. Soc. Japan *49*, 2042 (1980a).
[5] Macrossan, M. N.: Diatomic Collision Models Used in the Monte Carlo Direct Simulation Method Applied to Rarefied Hypersonic Flows. Ph. D. Thesis, University of London 1983.
[6] Ploss, H.: Simulationsmethoden zur Lösung der Boltzmanngleichung. Bericht der Arbeitsgruppe Technomathematik Nr. 11, Universität Kaiserslautern 1985.
[7] Babovsky, H.: On a simulation scheme for the Boltzmann equation. Math. Meth. Appl. Sci. *9*, to appear (1986).
[8] Nanbu, K.: Analysis of cylindrical Couette flows by use of the direct simulation method. Phys. Fluids *27*, 2632 (1984).
[9] Gentzsch, W.: Vectorization of Computer Programs with Applications to Computational Fluid Dynamics. Braunschweig; Wiesbaden: Vieweg 1984.
[10] Nanbu, K., Watanabe Y.: Analysis of the Internal Structure of Shock Waves by Means of the Exact Direct-Simulation Method. Rarefield Gas Dynamics, 14th Symposium, p. 183. University of Tokyo Press 1984.
[11] Alsemeyer, H.: Density profiles in Argon and nitrogen shock waves measured by the absorption of an electron beam. J. Fluid Mech. *74*, 497 (1976).

Dr. H. Ploss
Interuniversitäres Forschungsinstitut
für Fernstudien
Studienzentrum Bregenz
Belruptstrasse 10
A-6900 Bregenz
Austria