TA 8l.

On the efficiency of simulation methods for the Boltzmann equation on parallel computers

J. Struckmeier, F.J. Pfreundt
AG Technomathematik
University of Kaiserslautern
Germany

Abstract

The paper presents a parallelization technique for the finite pointset method, a numerical method for rarefied gas flows.
First we give a short introduction to the Boltzmann equation, which describes the behaviour of rarefied gas flows. The basic ideas of the finite pointset method are presented and a strategy to parallelize the algorithm will be explained. It is shown that a static processor partition leads to an insufficient load-balance of the processors. Therefore an optimized parallelization technique based on an adaptive processor partition will be introduced, which improves the efficiency of the simulation code over the whole region of interesting flow situation. Finally we present a comparison of the CPU-times between a parallel computer and a vector computer.

Keywords

Rarefied gas flows, Boltzmann equation, finite pointset method, parallelization technique, speedup factor, static processor partition, insufficient load-balance, adaptive processor partition, optimized load balance, comparison with vector computers.

1 Introduction

The prediction of aerodynamic coefficients during the reentry phase of a space vehicle is an important part in the design of such vehicles. In addition to windtunnel measurements numerical methods for the calculation of rarefied gas flows become a more and more efficient tool for such a prediction.
Compared to measurements the effort to build up computer programs including variable geometries is rather low. The only limit for such applications is the computational effort, mainly if it is necessary to include more physical phenomena, like vibrational energies or chemical reactions. During the last years a lot of effort was spent to optimize the performance of such numerical methods by using the vector facilities of supercomputers. Unfortunately the effective use of the vector facilities requires data vectors of a big length; a situation, which occurs only in a small part of the whole method. As a consequence the real MFLOP-rate is always far away from the peak performance value of the supercomputer.
In the following we will study the efficiency of the finite pointset method for rarefied gas flows on a parallel computer, which seems to be much more suitable for such applications. The connection between the single processors is given by a hypercube architecture.
It will be shown, that the speed-up factor is near to the theoretical limit. Consequently even small parallel machines are faster than vector computers.
First we introduce the mathematical problem, i.e. the Boltzmann equation, and describe shortly the finite pointset method used for the calculations. A detailed description can be found in [5].
After that we explain the strategy to parallelize the method by dividing the spatial domain into subdomains connected to a single node in the hypercube. Furthermore we discuss a method to obtain a reasonable load balance between the processors, which influence mostly the speedup factor.
Finally we present the results on a nCUBE 2 parallel computer for a classical test example in rarefied gas dynamics and compare the computational times with the results on a vector computer.

2 The finite pointset method for the Boltzmann equation

The behaviour of rarefied gas flows is described by the Boltzmann equation, a highly nonlinear transport equation, which was formulated in the 1870's by Ludwig Boltzmann. In contrast to the equations of continuum flow, the Boltzmann equation is based on a microscopic treatment of the gas ensemble. The equation prescribes the evolution of the velocity distribution of a gas ensemble in space and time, at which the free movement of the gas particles is influenced by binary collisions.
We will consider in the following the Boltzmann equation for a simple, i.e. monoatomic, gas bounded in the spatial domain $\Omega \subset \mathbb{R}^3$.
In this case, the Boltzmann equation can be written as

$$\frac{\partial f}{\partial t} + v\nabla_x f = \frac{1}{\epsilon} J(f,f) \tag{1}$$

$$J(f,f) := \int_{\mathbb{R}^3} \int_{S^2_+} \sigma(|v-w|,\eta)\{f(t,x,v')f(t,x,w') - f(t,x,v)f(t,x,w)\}dw(\eta)dw \tag{2}$$

$$\overset{o}{f}(t,x,v)|_{t=0}^{o} = \overset{o}{f}(x,v) \tag{3}$$

$$|v\cdot n| f(t,x,v) = \int_{v'\cdot n<0} R(v' \to v;t,x)|v'\cdot n| f(t,x,v')dv' \tag{4}$$

$$\text{with} \quad v\cdot n > 0 \quad \text{and} \quad x \in \partial\Omega$$

The left side of equation (1) represents the free movement of the gas particles, which is disturbed by the collision operator $J(f,f)$.
The parameter ϵ in front of $J(f,f)$ is proportional to the mean free path of the gas ensemble and gives a weight between the free flow and the collisions.
In the limit $\epsilon \to \infty$ the gas is collisionfree, the so-called free molecular limit, where only the boundary $\partial\Omega$ can change the velocity distribution of the particles. For $\epsilon \to 0$ the equation passes into the continuum limit, which means that the collisions occur at a timescale, which is much smaller than the macroscopic changes; the gas can be considered as a gas at equilibrium. In this case it is possible to switch to the continuum equations, like Navier-Stokes - or Euler equations.
The collision operator $J(f,f)$ depends on the collision cross section $\sigma(|v-w|,\eta)$, a function, which defines the interaction potential of a binary collision. The most common used model for simple gases is to consider the gas atoms as hard spheres, which seems to be a sufficient assumption at high Mach numbers. In this case the function σ is given by

$$\sigma(|v-w|,\eta) := C\cdot|v-w|, \tag{5}$$

where C is related to the diameter of the atoms.

The system is closed by an initial condition $\overset{o}{f}(x,v)$ at time $\overset{o}{t}= 0$ and a boundary condition at the boundary $\partial\Omega$. The condition at $\partial\Omega$ is determined by a scattering kernel $R(v' \to v;t,x)$, which defines the velocity distribution of the outgoing particles at a point $x \in \partial\Omega$ dependend on the incoming mass flux. There exists a lot of different scattering kernels, which are more or less sufficient to describe the exact gas-surface interaction. The one used for the current calculations is the complete accommodation with diffuse reflection: the velocity distribution of the outgoing particles is given by a halfspace maxwellian distribution based on a fixed wall temperature; the scattering kernel $R(v' \to v;t,x)$ is given by the formula

$$R(v' \to v; t, x) := |v \cdot n| f_w(v) \tag{6}$$

where f_w is a halfspace Maxwellian of the form

$$f_w(v) := [2\pi(RT_w(t,x))^2]^{-1} \exp(-v^2(2RT_w(t,x))^{-1})$$

and

$$T_w(t,x) \text{ the temperature at } x \in \partial\Omega.1cm$$

The Boltzmann equation represents a highly nonlinear, high dimensional transport equation; hence it is necessary to attack the equation by a numerical method. An additional difficulty is the fact that numerical methods applied usually in the case of partial differential equations are not suitable for the Boltzmann equation. The reason for this is that the distribution function f depends on at least seven variables and the collision operator on the right hand side of the equation is given by a fivefold integral term. As a consequence it is necessary to introduce other kinds of approximations to calculate solutions of the equation.

The idea of the finite pointset method is to use a set of discrete points in the phase space $\Omega \times \mathbb{R}^3$, which approximate the function $f(t,x,v)$ at a given time t; an idea which was introduced by Neunzert and Wick in the case of the Vlasov-equation. The changes in the set of the discrete points are subjected to a rule, which is given by the time evolution of the function f and thereby by the Boltzmann equation. This kind of approximation is mathematically based on the approximation of an absolutely continuous probability measure μ on $\Omega \times \mathbb{R}^3$ by a sum of discrete measures μ_N in the form

$$\mu_N := \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i) \times \delta(v - v_i) \underset{N \to \infty}{\overset{w}{\to}} f dv dx, \tag{7}$$

where f is the density of the measure μ and "w" denotes the weak convergence. It is out of the scope of the paper to give a detailed description of the finite pointset method, especially of the mathematical tools to build up the method; hence we will give in the following a short description how to construct an approximating pointset and subsequently the algorithm of the finite pointset method. A complete description of the method can be found in [5].

Let μ be an absolutely continuous probability measure on $\mathbb{R}$ with density f.

Then it holds that

$$f(t) \geq 0 \quad \forall_{t \in \mathbb{R}}, \tag{8}$$

$$\int_{\mathbb{R}} f(t)dt = 1 \tag{9}$$

and

$$\int_{-\infty}^{x} f(t)dt = z \in [0,1] \tag{10}$$

Now we take a finite sequence $\{z_i\}_{i=1,\dots,N}$ of uniformly distributed points in $[0,1]$.
By formula (10) we can calculate the corresponding set of points $\{x_i\}_{i=1,\dots,N} \subset \mathbb{R}$, i.e.

$$\int_{-\infty}^{x_i} f(t)dt = z_i \quad \forall_{i=1,\dots,N}. \tag{11}$$

It can be shown that $\{x_i\}_{i=1,\dots,N}$ is uniformly distributed on $\mathbb{R}$ with respect to the probability measure μ. With this pointset $\{x_i\}_{i=1,\dots,N}$ we can calculate moments of the measure μ by the formula

$$\frac{1}{N}\sum_{i=1}^{N} \varphi(x_i) \underset{N\to\infty}{\longrightarrow} \int \varphi(t)f(t)dt \tag{12}$$

and the relation holds for all testfunction φ, if the points $\{x_i\}_{i=1,\dots,N}$ are determined by the formula (11). Further it is possible to give an error estimate for a finite number N, which is based on the quality of the uniformly distribution of the points $\{z_i\}_{i=1,\dots,N}$.
This technique is used for the numerical solution of the Boltzmann equation:
The initial condition $\overset{\circ}{f}(x,v)$ is approximated by a finite pointset $\{\overset{\circ}{x}_i, \overset{\circ}{v}_i\}_{i=1,\dots,N}$ in the same way as given by formula (11).
Then the task is to perform a time evolution of the points $\{\overset{\circ}{x}_i, \overset{\circ}{v}_i\}_{i=1,\dots,N}$, such that the set $\{x_i(t), v_i(t)\}_{i=1,\dots,N}$ is an approximation of the solution of the Boltzmann equation on the time intervall $[0,T]$.
To get the rule for the time evolution of the points the Boltzmann equation is discretized in space and time:
Let $\{\Omega_i\}$ be a disjoint covering of the spatial domain, i.e.

$$\Omega = \overset{\circ}{\bigcup_{i=1,\dots,K}} \Omega_i \tag{13}$$

and Δt a discrete timestep on the interval $[0,T]$, i.e.

$$\Delta t = \frac{T}{L} \tag{14}$$

Now we consider the time- and space-discretized Boltzmann equation

$$\tilde{f}(\Delta t, x, v) = (1 + \frac{\Delta t}{\epsilon} J)(P\tilde{f})(x,v) \tag{15}$$

where

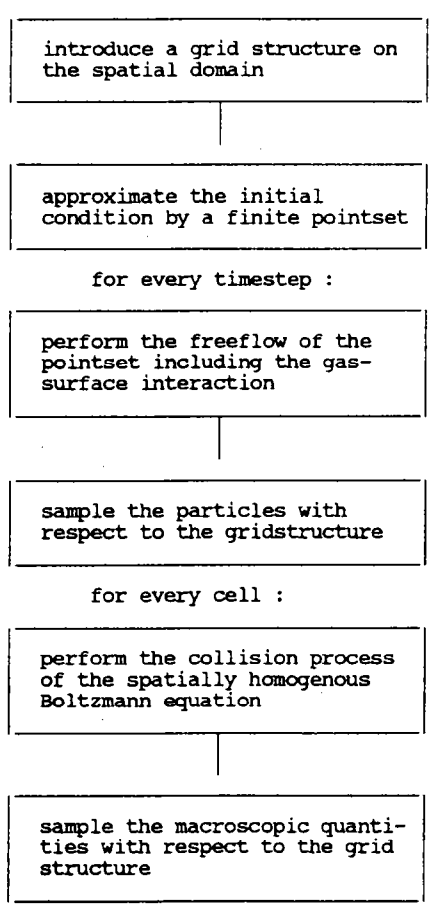
$$\tilde{f}(\Delta t, x, v) := \sum_{i=1}^{K} f_{\Omega_i}(\Delta t, v) \tag{16}$$

and

$$(P\tilde{f})(x,v) := \frac{\int_{\Omega_i} \tilde{f}(0, x - v\Delta t, v)dx}{Vol(\Omega_i)} \qquad \text{for } x \in \Omega_i \tag{17}$$

The distribution function $f(t,x,v)$ is replaced by a step function on the basis of the introduced grid structure; therefore it is necessary to consider the operator P, which projects to the set of step functions on $\{\Omega_i\}_{i=1,\ldots,K}$. By the discretized equation (15) one can recognize the rule for the time evolution of the pointset $\{\overset{\circ}{x}_i, \overset{\circ}{v}_i\}_{i=1,\ldots,N}$:

The free flow of the particles and the collision process are decoupled by the timestep Δt; further the collision operator J is applied to the step function $\tilde{f}$ on $\{\Omega_i\}_{i=1,\ldots,u}$, which means that in each cell Ω_i the collision procedure is based on the spatially homogenous Boltzmann equation.
The algorithm can be summarized by the following diagram:



The macroscopic quantities of the flow, like density or temperature, are calculated by using the step function $\tilde{f}$ analogous to formula (12):

$$\varphi_{\Omega_i} := \frac{1}{N} \sum_{i=1}^{N} \chi_{\Omega_i}(x_i)) \varphi(v_i) \tag{18}$$

where χ_{Ω_i} denotes the characteristic function on Ω_i.
The finite pointset method is actually used to calculate the reentry conditions of the European space shuttle HERMES at altitudes, where it is necessary to consider the Boltzmann equation for the description of the flow situation. In references [1] and [2] the reader can find, as examples, calculations, which were performed by the above described method.

3 The parallelized version of the finite pointset method

In the following section we describe the parallelization of the finite pointset method on the basis of a subdivision of the spatial domain.
To illustrate the strategy we will consider a classical threedimensional flow problem: the flow around a flat disc with angle of attack as shown in figure 1.

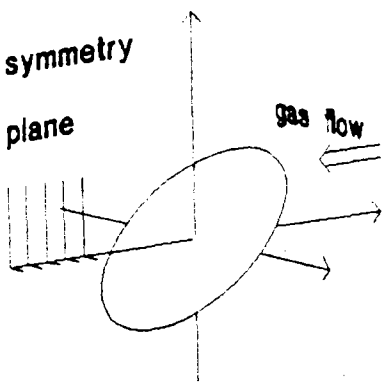


Fig. 1: Flat disc with angle of attack

In this flow problem one is mainly interested in the aerodynamic characteristics, like drag- or heat transfer coefficient, because this quantities can be measured with a high accuracy in a windtunnel (see [3]). However, to compare the calculations with measured data it is important to recognize, that the aerodynamic coefficients depend strongly on the scattering kernel used to model the gas-surface interaction. For this testcase the perfect accommodation with diffuse reflection, given by formula (6) shows a reasonable agreement with the measurements, except for the lift coefficient. The comparison between the calculation and the experiment will be given in the fourth chapter.
As mentioned above it is necessary to introduce a grid structure on the spatial domain,

used to perform collisions and to sample the macroscopic quantities. The cell system used for the actual computations consists of small cubes with a length smaller than the mean free path of the undisturbed gas (figure 2). Hereby the symmetry of the flowfield is taken into account. The flat disc is centered at the origin with an angle of attack measured with respect to the x-z-plane; the incoming gas flow is parallel to the x-direction. The main parallel property of the finite pointset method is the collision procedure: collisions can occur only for particle pairs, that are located in the same cell. As a consequence the collision procedure in a single cell is independent of all others in different cells. In addition the collision process is the most time consuming part in the whole algorithm. It is obvious to use this fact as a strategy to parallelize the simulation method. In dependence on the number of processors used during the computation, each processor is responsible for a certain number of cells to calculate the approximating pointset.

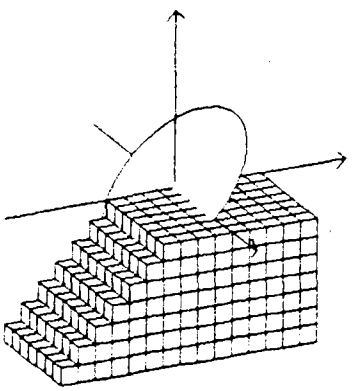


Fig. 2: Illustration of the grid structure

Introducing such a partition one processor has to communicate with the others, if particles leave the actual processor domain during the free flow or the gas-surface interaction. Consequently it is necessary to minimize the number of particles, that cross the processor boundaries during one timestep and with that the communication times between the single processors.
This is obtained by a partition as shown in figure 3:
cells, which are located in the x-direction, are combined to one stick and each processor is responsible for several sticks. This division considers on the one hand the incoming gas flow in the x-direction (most of the particles move mainly with the stream velocity) on

the other hand the scattering of particles at the flat disc.

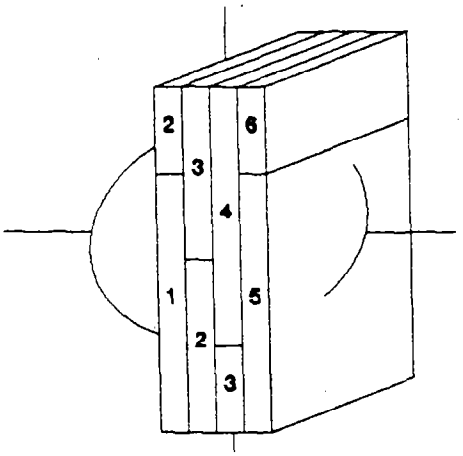


Fig. 3: Spatial processor partition

Unfortunately, with this static partition, fixed at the beginning of the computation, it is not possible to control the load-balance of the processors: the number of points in the different processor regions, which determines the load of a single processor, can be very different. The different particle numbers results in an insufficient load-balance and this fact is reflected by the speedup factor on a 32-node machine as shown in table 1. Since it is not possible to run the threedimensional code on a single processor the speed up factors in this paper are based on a theoretical value for the CPU-time on a single mode.
The estimate is given by

$$\sum_{i=1}^{n}(t_i - p_i), \tag{19}$$

where n is the number of processors in the hypercube, t_i the total CPU-time on the i-th node and p_i the time for the communication and the additional arithmetic operations, which are necessary to run the program on a parallel machine. Then the speedup factor is given by

$$\sum(t_i - p_i)/\max_{i=1,\ldots,n} t_i. \tag{20}$$

As shown in table 1 the speedup factors depend on the rarefactness of the flow (i.e. the Knudsen-number) and further on the angle of attack of the flat disc. Figure 4 shows the speedup factors in dependence on the Knudsen number for different angles of attack, figure 5 the speedup factor in dependence on the angle of attack for different Knudsen numbers.

Kn	angle	1-node CPU[s]	32-node CPU[s]	Speedup
0.05	45	14089.2	609.4	23.1
	60	14477.2	589.8	24.5
	75	14779.9	571.4	25.8
	90	14860.6	562.5	26.4
0.1	45	10454.0	451.5	23.1
	60	10697.5	436.5	24.5
	75	10876.2	421.6	25.7
	90	10918.4	414.4	26.3
0.2	45	10205.9	434.8	23.4
	60	10393.1	419.4	24.7
	75	10522.3	400.1	26.2
	90	10550.4	393.5	26.8
0.5	45	9829.1	398.5	24.6
	60	9956.8	386.5	25.7
	75	10034.1	369.0	27.1
	90	10048.6	353.5	28.4
1.0	45	9465.3	352.0	26.8
	60	9532.5	350.4	27.1
	75	9567.7	347.0	27.5
	90	9639.5	340.6	28.3
2.0	45	6476.0	248.5	26.0
	60	6507.6	246.6	26.3
	75	6525.9	243.5	26.8
	90	6535.0	239.7	27.2
5.0	45	6437.0	250.3	25.7
	60	6478.4	246.1	26.3
	75	6506.2	242.9	26.7
	90	6505.1	240.3	27.0

Tab. 1: CPU-times on a nCUBE 2 computer with static processor partition

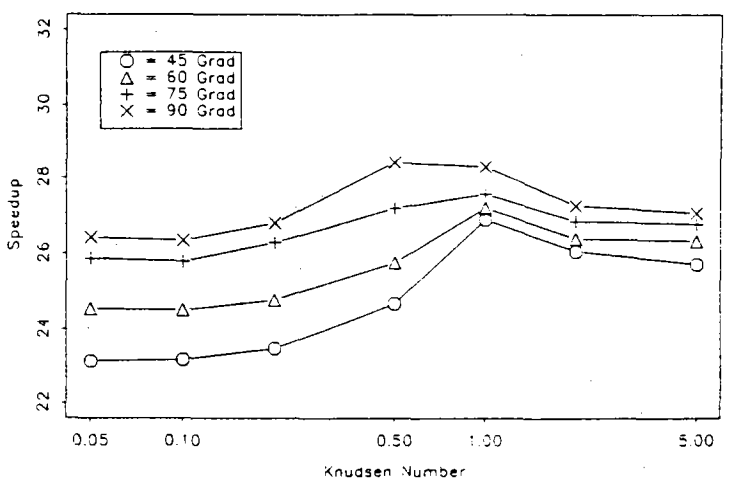


Fig. 4: Speedup factor vs. Knudsen number at different angles of attack

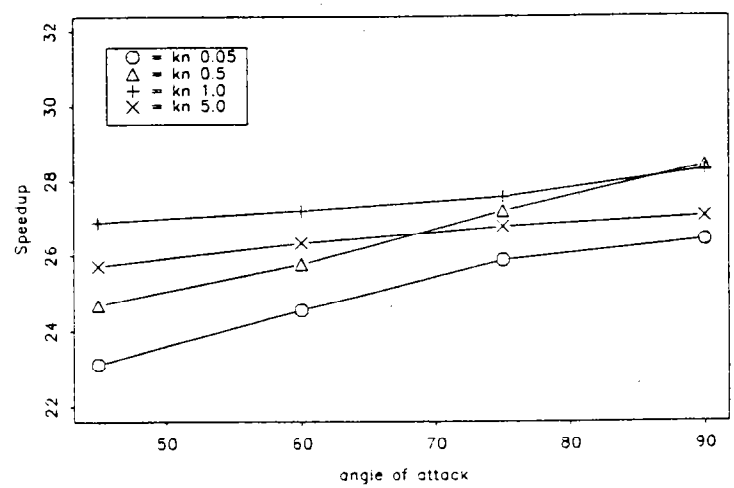


Fig. 5: Speedup factor vs. angles of attack at different Knudsen numbers

The number of particles on the different nodes is given in figure 6. This picture corresponds to the communication times on the single nodes as shown in figure 7. The strong dependence of the load-balance on the particle number is obvious. To get a better load-balance on the processors and with that a higher speedup factor over the whole flow regime it is necessary to develop an adaptive method to distribute the points of the computational domain more uniformly on the processors within the computation. Such an optimized parallelization technique will be given in the next section.

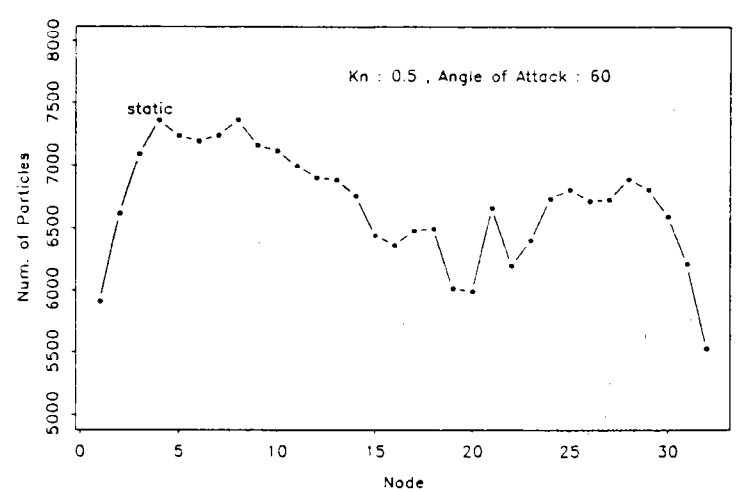


Fig. 6: Number of simulation particles on the single processors

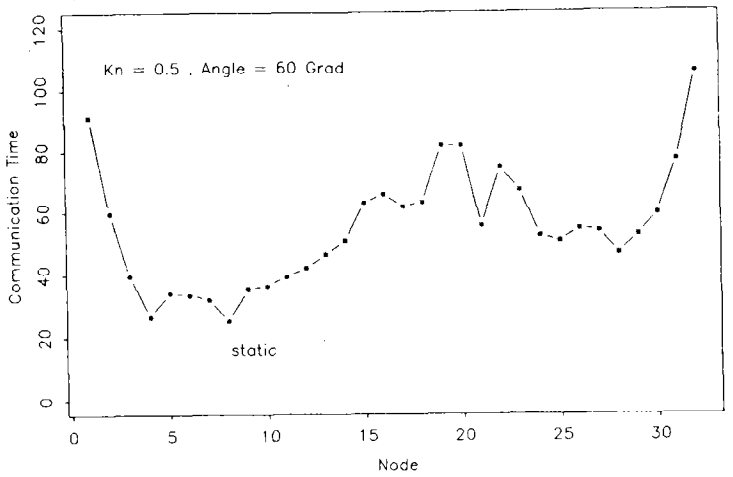


Fig. 7: Communication times of the single processors

4 An optimized parallelization technique

In the last section it was shown, that a static processor partition leads to an insufficient load-balance and therefore to a reduced speedup factor of the parallel code. To increase the efficiency it is necessary to distribute the approximation points more uniformly on the different nodes. There exists a lot of possibilities to obtain such an optimized distribution. A very simple, but nevertheless sufficiently accurate and fast method to perform the adaptive balancing is to exchange spatial sticks of the minimal and maximal loaded proccssors. By reiterating the exchange up to a reasonable error bound the spatial sticks can be distributed on the nodes in such a way, that each processor handles (up to a certain error) the same number of particles.
By this procedure the local behaviour of the communication between the processor is destroyed, but the improved load-balance overcomes this lack. In detail the adaptive balancing can be described by the following algorithm:

Let

	N_p	be the number of processors in the hypercube,
	N_s	the total number of spatial sticks
		(for simplicity we assume that N_s/N_p is an integer)
	M_K	the number of particles in the k-th stick
and	π_{ij}	the index of the i-th stick connected to processor j

Then

$$N_j := \sum_{i=1}^{N_s/N_p} M_{\pi_{ij}} \tag{21}$$

is the number of points connected to the processor j and

$$N := \sum_{j=1}^{N_p} N_j \tag{22}$$

the total number of particles in the spatial domain.
The indices j_{min} and j_{max} of the minimal and maximal loaded processors are given by

$$N_{j_{min}} := \min_{j=1,\ldots,N_p} N_j \tag{23}$$

and

$$N_{j_{max}} := \max_{j=1,\ldots,N_p} N_j. \tag{24}$$

We will achieve an optimal load-balance, if the number of particles on a single processor is equal to (or near to) the averaged number N^{opt}, where N^{opt} is given by

$$N^{opt} := N/N_p \tag{25}$$

For the processors j_{min} and j_{max} we can define an error $E(\{\pi_{ij_{min}}\}, \{\pi_{kj_{max}}\})$ on the basis of an optimal load-balance by

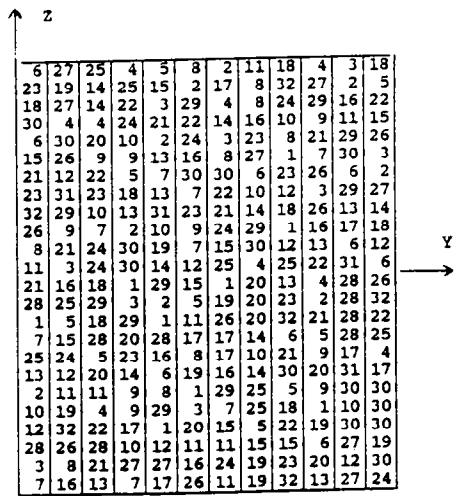
$$E(\{\pi_{ij_{min}}\}, \{\pi_{kj_{max}}\}) := |N^{opt} - N_{j_{min}}| + |N^{opt} - N_{j_{max}}|. \tag{26}$$

Now we try to minimize the functional E by a simple binary exchange of sticks of processor j_{min} and j_{max}:

$$\begin{aligned} E_{min} := \min_{\substack{i=1,\ldots,N_s/N_p \\ k=1,\ldots,N_s/N_p}} \{ & |N^{opt} - N_{j_{min}} + M_{\pi_{ij_{min}}} - M_{\pi_{kj_{max}}}| \\ & + |N^{opt} - N_{j_{max}} + M_{\pi_{kj_{max}}} - M_{\pi_{ij_{min}}}| \} \end{aligned} \tag{27}$$

By exchanging the sticks, which define the extremal value of E in (27), we improve the load-balance of the minimal and maximal loaded processor.
This procedure is repeated several times starting with the calculation of j_{min} and j_{max} given by (23) and (24) as long as the difference $N_{j_{max}} - N_{j_{min}}$ exceed a suitable error bound ϵ (i.e. $(N_{j_{max}} - N_{j_{min}})/N < \epsilon$).
Each optimization step leads to an optimal exchange between the current minimal and

maximal particle numbers. This simple algorithm yields to a very effective load-balance, at the same time the effort to perform this procedure is very low.



6	27	25	4	5	8	2	11	18	4	3	18
23	19	14	25	15	2	17	8	32	27	2	5
18	27	14	22	3	29	4	8	24	29	16	22
30	4	4	24	21	22	14	16	10	9	11	15
6	30	20	10	2	24	3	23	8	21	29	26
15	26	9	9	13	16	8	27	1	7	30	3
21	12	22	5	7	30	30	6	23	26	6	2
23	31	23	18	13	7	22	10	12	3	29	27
32	29	10	13	31	23	21	14	18	26	13	14
26	9	7	2	10	9	24	29	1	16	17	18
8	21	24	30	19	7	15	30	12	13	6	12
11	3	24	30	14	12	25	4	25	22	31	6
21	16	18	1	29	15	1	20	13	4	28	26
28	25	29	3	2	5	19	20	23	2	28	32
1	5	18	29	1	11	26	20	32	21	28	22
7	15	28	20	28	17	17	14	6	5	28	25
25	24	5	23	16	8	17	10	21	9	17	4
13	12	20	14	6	19	16	14	30	20	31	17
2	11	11	9	8	1	29	25	5	9	30	30
10	19	4	9	29	3	7	25	18	1	10	30
12	32	22	17	1	20	15	5	22	19	30	30
28	26	28	10	12	11	11	15	15	6	27	19
3	8	21	27	27	16	24	19	23	20	12	30
7	16	13	7	17	26	11	19	32	13	27	24

Fig. 8: Final state of the adaptive processor partition

Figure 8 shows, as an example, the final partition of the processors in the spatial domain. The processors seem to be uniformly distributed on the spatial domain; nevertheless the communication times increase just by a small amount, although the local behaviour of the communication has been destroyed.

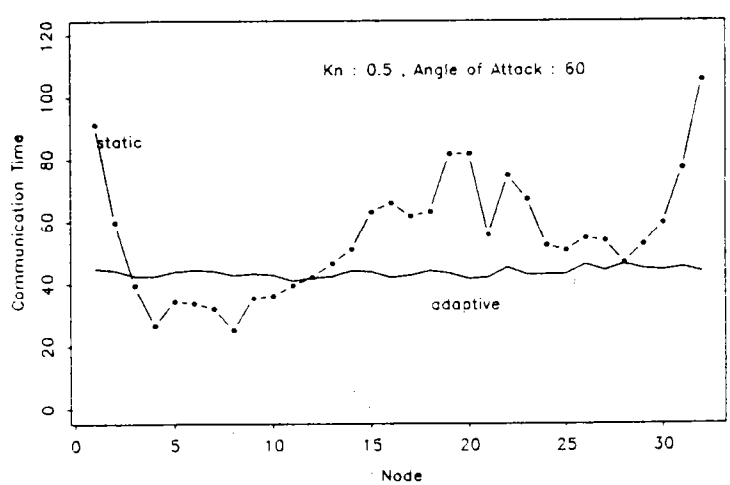


Fig. 9: Communication times of the single processors

Figure 9 shows the communication times of the single processors, figure 10 the corresponding particle number.

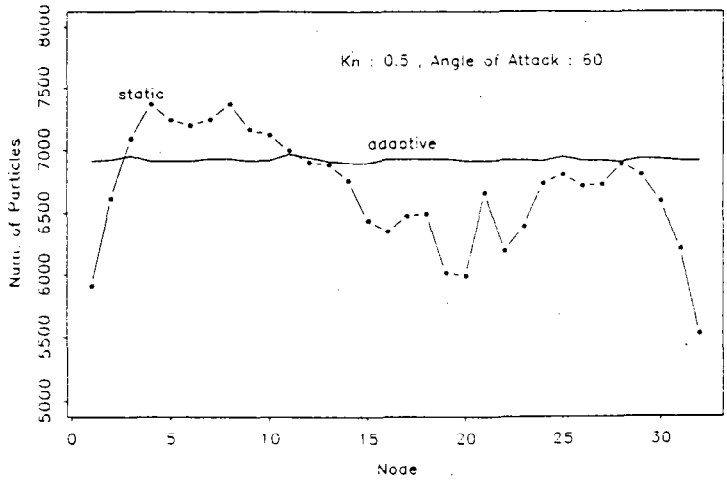


Fig. 10: Number of simulation particles on the single processors

The particle numbers are nearly the same on all processors. The improved load-balance can be easily verified by the CPU-times of the optimized simulation code, as given in table 2, and in more detail by figure 11 and 12.

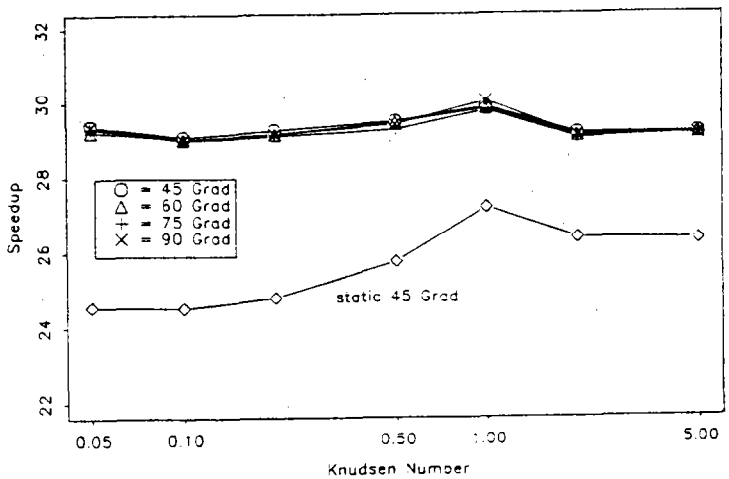


Fig. 11: Speedup factor vs. Knudsen number at different angles at attack

Kn	angle	1-node CPU[s]	32-node CPU[s] adaptive	Speedup static	adapt
0.05	45	14089.2	479.5	23.1	29.3
	60	14477.2	495.3	24.5	29.2
	75	14779.9	504.0	25.8	29.3
	90	14860.6	506.2	26.4	29.3
0.1	45	10454.0	359.5	23.1	29.0
	60	10697.5	368.6	24.5	29.0
	75	10876.2	375.2	25.7	29.0
	90	10918.4	376.3	26.3	29.0
0.2	45	10205.9	348.9	23.4	29.2
	60	10393.1	356.6	24.7	29.1
	75	10522.3	361.6	26.2	29.1
	90	10550.4	362.0	26.8	29.1
0.5	45	9829.1	333.3	24.6	29.5
	60	9956.8	338.0	25.7	29.5
	75	10034.1	342.9	27.1	29.3
	90	10048.6	341.2	28.4	29.4
1.0	45	9465.3	317.0	26.8	29.8
	60	9532.5	319.7	27.1	29.8
	75	9567.7	321.4	27.5	29.8
	90	9639.5	321.1	28.3	30.0
2.0	45	6476.0	222.1	26.0	29.1
	60	6507.6	223.7	26.3	29.0
	75	6525.9	224.8	26.8	29.0
	90	6535.0	224.6	27.2	29.0
5.0	45	6437.0	220.7	25.7	29.1
	60	6478.4	222.1	26.3	29.1
	75	6506.2	223.3	26.7	29.1
	90	6505.1	223.4	27.0	29.1

Tab. 2: CPU-times on a nCUBE 2 computer with adaptive processor partition

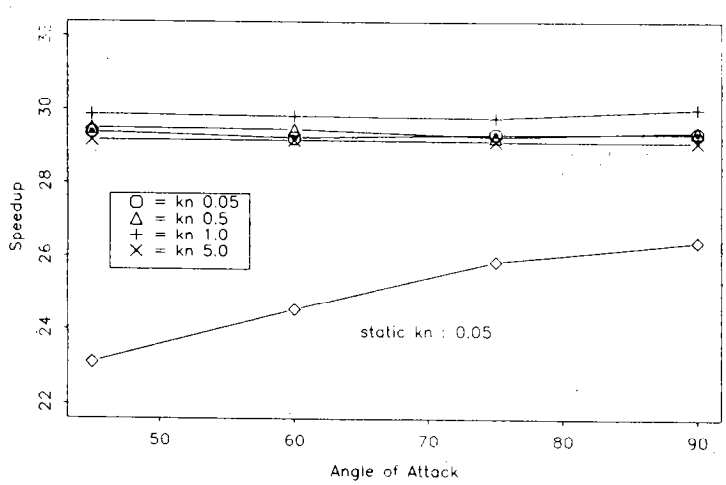


Fig. 12: Speedup factor vs. angle of attack at different Knudsen numbers.

The figures show the speedup factor of the the parallelized code, first versus the Knudsen number, second versus the angle of attack. In contrast to figure 4 and 5 the speedup factor of the optimized version remains nearly stable over the whole region at a value of about 29.5, which corresponds to an improvement of about 30 % in the maximal case.
The above described adaptive processor partition can be applied straight forward with the same efficiency to other flow problems in rarefied gas dynamics, because the parallelization strategy is independent of the geometrical data of the model object.
Furthermore the parallelized program can be designed independent of the number of processors as long as the number of spatial sticks is greater than the number of processors. Finally, the code remains nearly the same, since the same run module can be loaded on each processor.
As mentioned in the introduction the MFLOP-rate of simulation methods on a vector computer is far away from the peak performance value, because the vector length in the time-consuming parts of the method is rather small. We also mentioned that a parallel architecture seems to be more suitable for such applications. This statement is proved in table 3, which shows a comparison of the CPU-times on different computers.

computer	peak MFLOP-rate	CPU-time in [s]	factor
nCUBE 2 parallel computer			
32 - nodes	96	333.3	1.0
16 - nodes	48	640.5	1.9
8 - nodes	24	1232.1	3.7
Fujitsu VP100 vector computer	285	1075.0	3.2

Tab. 3: CPU-times at Kn = 0.5 and 45 degree angle of attack

Although the peak performance value of the vector computer VP 100 is nearly a factor 3 higher than the value of a 32-node nCUBE 2 parallel computer, the parallelized version of the finite pointset method has a better real performance.
Finally we present briefly a comparison between the calculated and the measured data. As mentioned in chapter 3 the most interesting quantities in the above described three dimensional testcase are the global aerodynamic characteristics. Figure 13 shows a comparison between the calculated drag coefficient (lines) and the measured data (dots), figure 14 the corresponding lift coefficient and finally figure 15 the heat transfer coefficient.

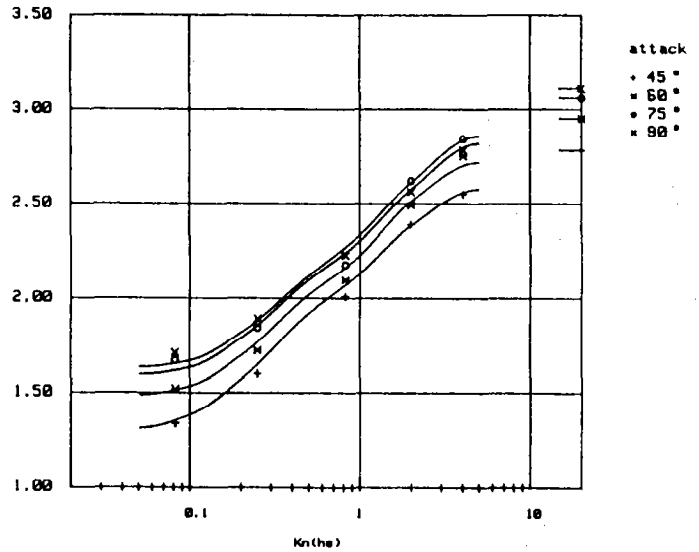


Fig. 13: Drag coefficient vs. Knudsen number

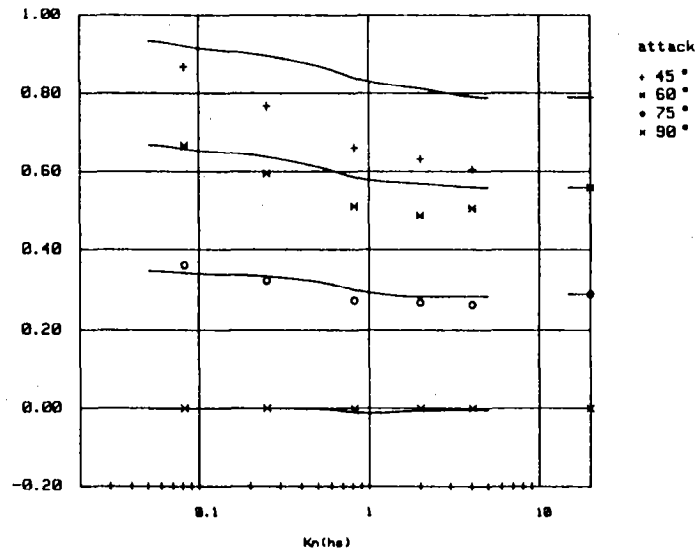


Fig. 14: Lift coefficient vs. Knudsen number

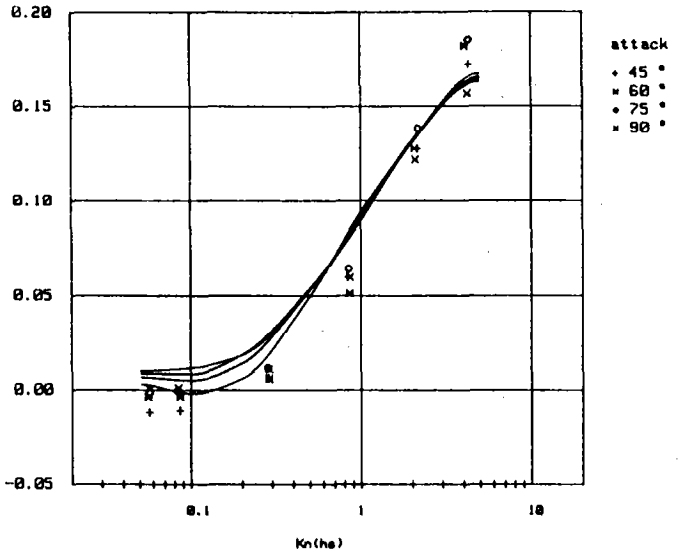


Fig. 15: Heat-transfer coefficient vs. Knudsen number

The agreement between the measurements and the calculations is reasonable within a small error bound except for the lift coefficient at small angles of attack and high knudsen number, a fact which was already recognized in reference [2].

5 Conclusion

As shown in this paper the finite pointset method turned out to be a very efficient numerical method on parallel computers: the achieved speedup factors are near to the theoretical limit. The strong parallelization is based on an adaptive control of the load-balance of the processors and this load-balance can be obtained by a simple optimization strategy. With this technique it is possible to obtain nearly constant speedup factors over the whole transition regime between free molecular and continuum flow. As a consequence the performance turned out to be better than on a vector computer, even on small parallel machines. With that it is possible to include more physical phenomena like chemical reactions within a reasonable computational effort. Finally, by using parallel computers, it will be possible to perform accurate calculations at the transition between rarefied and continuum flow.

Acknowledgement

The authors would like to thank Prof. Nelson and his research group for their kind hospitality at the Texas A&M University, College Station, and the possibility to perform calculations on the nCUBE 2 parallel computer; further the people from nCUBE Germany, especially H. Strauss, who contribute to a successful implementation of the program code

References

[1] Gropengiesser, F., Neunzert, H., Struckmeier, J., Wiesen, B.: *Hypersonic flow calculations around a 3d-deltawing at low Knudsen numbers*, in Beylich, A. (Ed.), Proc. of the 17th Int. Symposium on Rarefied Gas Dynamics, Aachen 1990, VCH Verlagsgesellschaft, Weinheim (1991), p. 332-336.

[2] Gropengiesser, F., Neunzert, H., Struckmeier, J., Wiesen, B.: *Rarefied gas flow around a disc with different angles of attack*, in: Beylich, A. (Ed.), Proc. of the 17th Int. Symposium on Rarefied Gas Dynamics, Aachen 1990, VCH Verlagsgesellschaft, Weinheim (1991), p. 546-553.

[3] Legge, H.: *Force and heattransfer measurements on a disc at* $45^o - 90^o$ *angle of attack in free jet flow using* Av, He, N_2, H_2 *as test gases*, DLR-report IB-222-89A07, Göttingen, 1989.

[4] Legge, H., Nanbu, K., Igarashi, S.: *Force and heat transfer on a disc in rarefied flow*, in: Beylich, A. (Ed.), Proc. of the 17th Int. Symposium on Rarefied Gas Dynamics, Aachen 1990, VCH Verlagsgesellschaft, Weinheim (1991), p. 679-686.

[5] Neunzert, H., Gropengiesser, F., Struckmeier, J.: *Computational methods for the Boltzmann equation*, in: Spigler, R. (Ed.), Applied and industrial mathematics, Venice-1, 1989, Kluver acad. publ., Dordrecht (1991) p. 111-140.