



Numerical aspects of the smoothed particle hydrodynamics method for simulating accretion disks

Harald Riffert, Heinz Herold, Olaf Flebbe, Hanns Ruder

Theoretische Astrophysik, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany

Abstract

The derivation of the Smoothed Particle Hydrodynamics (SPH) method is reviewed. In particular, the problem of second-order derivative terms is investigated. Applying these considerations to the Navier–Stokes equations, a physical viscosity is constructed which can be used to perform simulations of viscous fluids within the framework of SPH. With such a viscous stress tensor the energy balance and the angular momentum conservation for the particle and the continuum representations are compared. An SPH code based on these results is tested on different problems, especially on an analytically solvable problem, namely the spreading of a ring of gas moving with Keplerian speed around a point mass. Additionally, some examples for the dynamics of accretion disks in close binary systems are presented. Finally, the efficient implementation of this SPH code is discussed in some detail, in particular by a comparison between scalar and vector computers.

1. Introduction

There are several different methods to solve the hydrodynamic equations numerically either using grid-based algorithms or particle methods. The basic idea behind the particle methods is that the fluid can be divided into cells – henceforth called particles – which move under the influence of external forces and interact with each other in order to simulate the hydrodynamic properties of the fluid. An advantage of these methods is that the continuity equation is automatically fulfilled.

A number of different techniques has been developed in the past. One approach of particle hydrodynamics called Particle In Cell (PIC) (Harlow 1964) describes the particle interaction by intervecating the particle properties onto a grid, solving the equations on the grid, and then intervecating back to the particles. A serious problem of this method is to ensure numerical stability of the inter- and extraveccations. Furthermore, one must know the location of the regions in space where steep gradients occur to make the grid fine enough to resolve these structures well.

Another more recent formulation – the one we are going to discuss – is the *Smoothed Particle Hydrodynamics* (SPH) where no grid is needed. The equations are solved by formulating the particle-particle interaction directly. The forces are intervecated from a grid of moving points, i.e. the particles themselves. Since the first papers on the SPH technique (Lucy 1977, Gingold and Monaghan 1977) a large amount of work has been done in the framework of this approach (for reviews on the technical aspects see Monaghan 1985, Benz 1990, Monaghan 1992).

Especially in the context of accretion disks – the application which we deal with in this paper – there are advantages of SPH compared with other hydrodynamical schemes which we want to emphasize: (i) Taking account of free boundaries is no problem. (ii) Large density contrasts originate from the vacuum-matter boundaries which is natural for SPH (no matter – no particles). (iii) The dynamics is dominated by gravity forces which is convenient for the particle description.

In Section 2 of this paper we give a general outline of the derivation of the SPH equations, where the formulation of the *smoothing* of function values and of their first and second derivatives is discussed. Additionally the *particle* evaluation of the integrals is described.

Section 3 is devoted to the application of these ideas to the hydrodynamical equations. One problem with particle methods is the description of viscosity. However, in a number of astrophysical situations, for example accretion disks, the corresponding physical properties are dominated by viscous processes. Therefore, we present a formulation of physical viscosity in the framework of SPH and derive an expression for the entropy production which allows for the computation of energy dissipation (cf. Flebbe et al. 1994). Additionally, the conservation of energy and angular momentum is discussed.

This method is tested in Section 4 for the case of the viscous spreading of a gas ring around a central mass. An astrophysical application is the simulation of accretion disks, especially in close binary systems, e.g., for cataclysmic variables.

Finally, in Section 5 we present some details on the efficient implementation of an SPH code for scalar and vector computers.

2. Fundamentals of the SPH method

One ingredient of the SPH technique is the smoothing procedure: For each physical quantity $f = f(\mathbf{r})$ (e.g., density, pressure etc.), considered as a function of the spatial coordinates \mathbf{r} , its smoothed value is defined by

$$\langle f(\mathbf{r}) \rangle = \int d\mathbf{r}' f(\mathbf{r}') W(\mathbf{r}, \mathbf{r}', h) \quad (1)$$

where $W(\mathbf{r}, \mathbf{r}', h)$ is an interveccating kernel, $f(\mathbf{r}')$ the interveccated function and $d\mathbf{r}'$ the volume element $d^2\mathbf{r}'$ for 2D or $d^3\mathbf{r}'$ for 3D calculations, respectively. From now on we assume that the kernel is translationally invariant and spherically symmetric, i.e. only depends on the distance $|\mathbf{r} - \mathbf{r}'|$ and is of the form $W(|\mathbf{r} - \mathbf{r}'|, h)$, which greatly simplifies the discussion. The parameter h is a measure of the width of the interveccation. Furthermore the kernel has to be normalized to one when integrated over the support of W

$$\int d\mathbf{r}' W(|\mathbf{r} - \mathbf{r}'|, h) = 1 \quad (2)$$

and should have the convergence property

$$\lim_{h \rightarrow 0} W(|\mathbf{r} - \mathbf{r}'|, h) = \delta(\mathbf{r} - \mathbf{r}') . \quad (3)$$

A Taylor expansion shows that the smoothing error produced by substituting $\langle f(\mathbf{r}) \rangle$ for $f(\mathbf{r})$ is of second order in h , i.e.

$$\varepsilon_h := \langle f(\mathbf{r}) \rangle - f(\mathbf{r}) = \text{const.}(\Delta f)h^2 + O(h^3) . \quad (4)$$

The essential point is that spatial derivatives can be calculated by a natural extension of the smoothing procedure of Eq. (1) as follows (we denote derivatives with respect to the α -component of \mathbf{r} or \mathbf{r}' by $_{,\alpha}$ or $_{,\alpha'}$):

$$\begin{aligned}
\langle f(\mathbf{r})_{,\alpha} \rangle &= \int d\mathbf{r}' f(\mathbf{r}')_{,\alpha'} W(|\mathbf{r} - \mathbf{r}'|, h) \\
&= - \int d\mathbf{r}' (f(\mathbf{r}') + g(\mathbf{r})) W(|\mathbf{r} - \mathbf{r}'|, h)_{,\alpha'} \\
&= \int d\mathbf{r}' (f(\mathbf{r}') + g(\mathbf{r})) W(|\mathbf{r} - \mathbf{r}'|, h)_{,\alpha}, \tag{5}
\end{aligned}$$

where we integrated by parts (ignoring surface terms) using the fact that W depends only on the relative vector $\mathbf{r} - \mathbf{r}'$. In this derivation a free integration constant $g(\mathbf{r})$ enters whose choice will be discussed later. Note that we have implicitly assumed that W is a C^1 kernel, i.e. W has continuous first derivatives. It is straightforward to show that for a sufficiently smooth function $f(\mathbf{r})$ the smoothing error is again of second order:

$$\langle f(\mathbf{r})_{,\alpha} \rangle = f_{,\alpha}(\mathbf{r}) + O(h^2). \tag{6}$$

It is possible to construct smoothed second derivatives in an analogous way by two integrations by parts – a C^2 kernel is then required – (Laguna 1994), but for the realization of a physical viscosity (see Section 3) we prefer to use the smoothing procedure twice as follows:

$$\begin{aligned}
\langle f(\mathbf{r})_{,\alpha\beta} \rangle &= \int d\mathbf{r}' f_{,\alpha}(\mathbf{r}') W_{,\beta}(|\mathbf{r} - \mathbf{r}'|, h) \\
&\approx \int d\mathbf{r}' \langle f_{,\alpha}(\mathbf{r}') \rangle W_{,\beta}(|\mathbf{r} - \mathbf{r}'|, h) \\
&= \int d\mathbf{r}' \int d\mathbf{r}'' f(\mathbf{r}'') W_{,\alpha}(|\mathbf{r}' - \mathbf{r}''|, h) W_{,\beta}(|\mathbf{r} - \mathbf{r}'|, h). \tag{7}
\end{aligned}$$

Thus, in expressions containing second derivatives we obtain a product of first derivatives of the smoothing kernel. Therefore, it is sufficient that the kernel W is only of class C^1 . As to the smoothing error, second-order accuracy is maintained, since it is not difficult to prove that

$$\langle f(\mathbf{r})_{,\alpha\beta} \rangle = f_{,\alpha\beta}(\mathbf{r}) + O(h^2). \tag{8}$$

The second ingredient of the SPH method is the evaluation of the integrals, which result from the smoothing operation, by discrete particle points. Thus, the integral in Eq. (1) is approximated by a sum

$$\int d\mathbf{r}' f(\mathbf{r}') W(|\mathbf{r} - \mathbf{r}'|, h) = \sum_j \frac{f_j}{n_j} W(|\mathbf{r} - \mathbf{r}_j|, h) + \varepsilon_N, \tag{9}$$

where $f_j := f(\mathbf{r}_j)$ is the value of f and n_j is the number density of particles at the position of particle j . Note that a second type of error appears here, namely the particle error ε_N which is related to the distribution of the particles. In principle, this error is large if the particles are randomly distributed (the particle sum is then a Monte Carlo estimate), but this is normally not the case, since the dynamical forces act on the particles in a systematic manner. If the particles form a regular crystal-like structure the particle errors are at the minimum. As discussed by Martin et al. (1993), for the total error it is best to have neither a random nor a regular, but a thermalized (relaxed) particle distribution.

Usually, instead of the number density n_i the mass density ρ_i defined by $\rho_i = n_i m_i$ is employed, where m_i denotes the mass of particle i . Inserting the mass density ρ for f in Eq. (9) and introducing the abbreviation $W_{ij} = W(|\mathbf{r}_i - \mathbf{r}_j|, h)$ leads to the relation

$$\langle \rho(\mathbf{r}) \rangle \Big|_{\mathbf{r}=\mathbf{r}_i} = \rho_i = \sum_j m_j W_{ij}, \tag{10}$$

which has an obvious interpretation as the smoothing of the particle masses in space according to the kernel W .

3. Formulation of hydrodynamics

The starting point for the simulation of a gas-dynamical system with SPH is the Navier–Stokes equation in Lagrangian formulation

$$\frac{dv_\alpha}{dt} = \frac{\partial v_\alpha}{\partial t} + v_\beta v_{\alpha,\beta} = -\frac{1}{\rho} p_{,\alpha} + \frac{1}{\rho} t_{\alpha\beta,\beta} + F_\alpha. \quad (11)$$

Apart from the last term on the right hand side which describes the external forces, the first term is the acceleration from the pressure gradient and the second expression contains the viscous stress tensor $t_{\alpha\beta}$ with the parameters η and ζ which are positive and do not depend on the velocity \mathbf{v} (Landau and Lifschitz 1991),

$$t_{\alpha\beta} = \eta (v_{\alpha,\beta} + v_{\beta,\alpha} - \frac{2}{3} \delta_{\alpha\beta} v_{\gamma,\gamma}) + \zeta v_{\gamma,\gamma}, \quad (12)$$

where the expression in the parentheses is the shear denoted by $\sigma_{\alpha\beta}$, i.e.

$$\sigma_{\alpha\beta} = v_{\alpha,\beta} + v_{\beta,\alpha} - \frac{2}{3} \delta_{\alpha\beta} v_{\gamma,\gamma}. \quad (13)$$

The energy dissipation per mass related to the viscous forces can be expressed in terms of derivatives of the velocity \mathbf{v}

$$T \frac{ds}{dt} = \frac{\eta}{2\rho} \sigma_{\alpha\beta} \sigma_{\alpha\beta} + \frac{\zeta}{\rho} (v_{\gamma,\gamma})^2, \quad (14)$$

where s is the entropy per mass.

For the sake of simplicity, we assume $\zeta = 0$ for the rest of this paper. However, including bulk viscosity is not a principal problem.

3.1. Smoothed particle discretization

In SPH the hydrodynamic Eq. (11) is approximated by taking the average of every quantity of interest, i.e. of the mass density, the pressure gradient and the viscous forces.

The rule expressed in Eq. (5) for the calculation of derivatives combined with the particle evaluation formula given in Eq. (9) can be immediately applied to the pressure gradient term in Eq. (11) at the position of particle i . This leads to (cf. Monaghan 1985)

$$\left\langle \frac{1}{\rho} p_{,\alpha} \right\rangle_i = \sum_j m_j \left(\frac{p_j + p_i}{\rho_i \rho_j} \right) (W_{,\alpha})_{ij}. \quad (15)$$

Here, a typical SPH approximation of the form

$$\left\langle \frac{1}{\rho} p_{,\alpha} \right\rangle_i \approx \frac{1}{\rho_i} \langle p_{,\alpha} \rangle_i \quad (16)$$

has been used. Additionally, in order to conserve linear and angular momentum on the particle level the force should be antisymmetric in the particles i and j ; obviously this has been achieved by choosing the integration constant $g(\mathbf{r})$ in Eq. (5) to be $f(\mathbf{r})$. The abbreviated notation

$$(W_{,\alpha})_{ij} = \frac{\partial}{\partial x_\alpha} W(|\mathbf{r} - \mathbf{r}'|, h) \Big|_{\mathbf{r}=\mathbf{r}_i, \mathbf{r}'=\mathbf{r}_j} \quad (17)$$

has been used here and will be applied in the following.

In order to build up the viscous force contained in Eq. (11), once more we combine Eqs. (5) and (9) and obtain (by choosing the integration constant of Eq. (5) such that the viscous force is antisymmetric in the particles i and j)

$$\left(\frac{dv_{\alpha i}}{dt}\right)_{\text{visc}} = \left\langle \frac{1}{\rho} t_{\alpha\beta,\beta} \right\rangle_i = \sum_j \frac{m_j}{\rho_i \rho_j} (\eta_j \sigma_{\alpha\beta j} + \eta_i \sigma_{\alpha\beta i}) (W_{,\beta})_{ij}. \quad (18)$$

Therein, the particle form $\sigma_{\alpha\beta i}$ of the shear from Eq. (13) is needed, which we write

$$\sigma_{\alpha\beta i} = \langle \sigma_{\alpha\beta} \rangle_i = V_{\alpha\beta i} + V_{\beta\alpha i} - \frac{2}{3} \delta_{\alpha\beta} V_{\gamma\gamma i}. \quad (19)$$

The quantity $V_{\alpha\beta i}$ is the particle equivalent of the velocity gradient $v_{\alpha,\beta}$ and is defined as

$$V_{\alpha\beta i} = \sum_k \frac{m_k}{\rho_k} (v_{\alpha k} - v_{\alpha i}) (W_{,\beta})_{ik}. \quad (20)$$

This is again a standard application of the ‘smoothing+particle’ procedure, but now the integration constant $g(\mathbf{r})$ of Eq. (5) has been chosen to be $-f(\mathbf{r})$ such the velocity difference $\mathbf{v}_k - \mathbf{v}_i$ appears. The reason for this choice is that the velocity gradient should vanish for a constant velocity field and as a consequence we have Galilean invariance. In summary, the representation of the viscous forces given by Eqs. (18)–(20) yields an expression which contains products of first derivatives of the kernel W and is equivalent to the treatment of second derivatives as discussed in Section 2 in connection with Eq. (7).

We want to stress the different intentions of this type of viscosity and of artificial viscosity as, e.g., described by Monaghan and Gingold (1983). Artificial viscosity is introduced to resolve shocks numerically and vanishes in the hydrodynamic limit $h \rightarrow 0$. This is good for shocks, but bad for flows with a real physical viscosity. Artificial viscosity is constructed in such a way that it is effective only in compression regions where $v_{\gamma,\gamma} < 0$. It is derived from a continuum formulation which contains no shear viscosity, although the particle form produces a shear as well as a bulk viscosity (Monaghan 1985).

3.2. Energy balance

The entropy production due to viscous dissipation is (cf. Eq. (14))

$$\rho T \frac{ds}{dt} = \frac{\eta}{2} \sigma_{\alpha\beta} \sigma_{\alpha\beta} = \eta \sigma_{\alpha\beta} v_{\alpha,\beta}. \quad (21)$$

In order to simplify the discussion we are omitting energy sources and sinks as well as external forces for the rest of this section.

Using the first law of thermodynamics the change of the internal energy per mass ε can be written

$$\frac{d\varepsilon}{dt} = T \frac{ds}{dt} - \frac{p}{\rho} v_{\alpha,\alpha}. \quad (22)$$

Combining this with the equation of motion one gets for the total energy

$$\frac{d}{dt} \left(\varepsilon + \frac{1}{2} v^2 \right) = -\frac{1}{\rho} (p v_{\alpha} - t_{\alpha\beta} v_{\beta})_{,\alpha}. \quad (23)$$

In order to investigate the energy balance for the particles, we discretize the entropy production (21) in the form

$$T_i \frac{ds_i}{dt} = \frac{\eta_i}{\rho_i} \sigma_{\alpha\beta i} V_{\alpha\beta i} \quad (24)$$

and introduce the internal energy per mass ε_i of particle i . Calculating its change by using (22), (24) and (20) we obtain

$$\frac{d\varepsilon_i}{dt} = T_i \frac{ds_i}{dt} - \frac{p_i}{\rho_i} V_{\alpha\alpha i} = \sum_j \frac{m_j}{\rho_i \rho_j} (v_{\alpha j} - v_{\alpha i}) (W_{,\beta})_{ij} (\eta_j \sigma_{\alpha\beta i} - p_i \delta_{\alpha\beta}) . \quad (25)$$

Adding the change in the kinetic energy per mass from the equation of motion

$$\begin{aligned} \frac{d}{dt} \left(\frac{1}{2} v_i^2 \right) &= v_{\alpha i} \frac{dv_{\alpha i}}{dt} \\ &= - \sum_j \frac{m_j}{\rho_i \rho_j} v_{\alpha i} (W_{,\beta})_{ij} ((p_j + p_i) \delta_{\alpha\beta} - (\eta_j \sigma_{\alpha\beta j} + \eta_i \sigma_{\alpha\beta i})) \end{aligned} \quad (26)$$

one gets

$$\begin{aligned} \frac{d}{dt} \left(\varepsilon_i + \frac{1}{2} v_i^2 \right) &= - \sum_j \frac{m_j}{\rho_i \rho_j} (W_{,\beta})_{ij} (p_i v_{\alpha j} + p_j v_{\alpha i}) \delta_{\alpha\beta} \\ &\quad + \sum_j \frac{m_j}{\rho_i \rho_j} (W_{,\beta})_{ij} (\eta_i \sigma_{\alpha\beta i} v_{\alpha j} + \eta_j \sigma_{\alpha\beta j} v_{\alpha i}) . \end{aligned} \quad (27)$$

Now we want to show that Eq. (27) can be written as the particle formulation of the total energy Eq. (23). Both the pressure and the viscous terms can be combined in the form

$$\sum_j \frac{m_j}{\rho_i \rho_j} (W_{,\beta})_{ij} (f_{\alpha\beta i} v_{\alpha j} + f_{\alpha\beta j} v_{\alpha i}) \quad (28)$$

with $f_{\alpha\beta i} = -p_i \delta_{\alpha\beta} + \eta_i \sigma_{\alpha\beta i}$. Substituting for $f_{\alpha\beta j}$ and $v_{\alpha j}$ the Taylor expansion at the point $\mathbf{r} = \mathbf{r}_i$ it can easily be proved that expression (28) is equal to

$$\sum_j \frac{m_j}{\rho_i \rho_j} (W_{,\beta})_{ij} (f_{\alpha\beta j} v_{\alpha j} + f_{\alpha\beta i} v_{\alpha i}) + O(h^2) \quad (29)$$

by using the width h of the intervecation kernel W as an estimate for $|\mathbf{r}_i - \mathbf{r}_j|$. Using this result we get for the total energy equation on the particle level

$$\begin{aligned} \frac{d}{dt} \left(\varepsilon_i + \frac{1}{2} v_i^2 \right) &= - \sum_j \frac{m_j}{\rho_i \rho_j} (W_{,\beta})_{ij} (p_j v_{\alpha j} + p_i v_{\alpha i}) \delta_{\alpha\beta} \\ &\quad + \sum_j \frac{m_j}{\rho_i \rho_j} (W_{,\beta})_{ij} (\eta_j \sigma_{\alpha\beta j} v_{\alpha j} + \eta_i \sigma_{\alpha\beta i} v_{\alpha i}) + O(h^2) . \end{aligned} \quad (30)$$

Obviously, this is the particle formulation of Eq. (23) using the SPH form of derivatives (with a well-defined integration constant).

One could have started with this form of the energy balance, but then it cannot be ensured that the entropy production on the particle level is positive. Therefore, we have preferred to use the positive definite form (24) of the energy dissipation as a direct discretization of Eq. (21).

3.3. Angular momentum conservation

We have constructed our particle formulation in such a way that, due to the antisymmetry of the forces, for the particles the momentum conservation is exactly fulfilled. As to angular momentum conservation, this

question will be considered in this section. First, in the continuum limit, the total angular momentum with components L_α is defined by

$$L_\alpha = \int d^3\mathbf{r} \varepsilon_{\alpha\beta\gamma} x_\beta \rho v_\gamma. \quad (31)$$

Therefore, using the momentum equation in the form

$$\frac{\partial}{\partial t} (\rho v_\alpha) = T_{\alpha\beta,\beta} \quad (32)$$

with the total stress tensor $T_{\alpha\beta} = -p\delta_{\alpha\beta} + t_{\alpha\beta}$, the temporal change of the angular momentum reads

$$\frac{dL_\alpha}{dt} = \int d^3\mathbf{r} \varepsilon_{\alpha\beta\gamma} x_\beta \frac{\partial}{\partial t} (\rho v_\gamma) = \int d^3\mathbf{r} \varepsilon_{\alpha\beta\gamma} x_\beta T_{\gamma\lambda,\lambda}. \quad (33)$$

Integration by parts reduces this expression to a surface integral

$$\frac{dL_\alpha}{dt} = \oint dA_\lambda \varepsilon_{\alpha\beta\gamma} x_\beta T_{\gamma\lambda} \quad (34)$$

due to the fact that the tensor $T_{\alpha\beta}$ is symmetric.

Transferring such an argument to the particle formulation yields

$$\begin{aligned} \frac{dL_\alpha}{dt} &= \sum_i \frac{dL_{\alpha i}}{dt} = \sum_i m_i \varepsilon_{\alpha\beta\gamma} x_{\beta i} \frac{dv_{\gamma i}}{dt} \\ &= \sum_{i,j} \frac{m_i m_j}{\rho_i \rho_j} \varepsilon_{\alpha\beta\gamma} (W_{,\lambda})_{ij} \left[x_{\beta i} T_{\gamma\lambda i} + x_{\beta j} T_{\gamma\lambda j} + (x_{\beta i} - x_{\beta j}) T_{\gamma\lambda j} \right] \\ &= \sum_{i,j} \frac{m_i m_j}{\rho_i \rho_j} \varepsilon_{\alpha\beta\gamma} (W_{,\lambda})_{ij} (x_{\beta i} - x_{\beta j}) T_{\gamma\lambda j}. \end{aligned} \quad (35)$$

To arrive at the last line the antisymmetry of $(W_{,\lambda})_{ij}$ in i and j has been exploited.

Since the gradient of the spherically symmetric kernel W lies in the direction of $\mathbf{r}_i - \mathbf{r}_j$, the pressure part of the stress tensor $T_{\gamma\lambda}$, which is proportional to $\delta_{\gamma\lambda}$, gives no contribution to the right hand side of Eq. (35). This is immediately understandable as the pressure force in the particle formulation is a central force.

As to the viscous stress tensor $t_{\gamma\lambda}$ in (35), it is necessary to approximate $t_{\gamma\lambda j}$ by making a Taylor expansion around $\mathbf{r} = \mathbf{r}_i$. This procedure yields

$$\frac{dL_\alpha}{dt} = \sum_i \frac{m_i}{\rho_i} \varepsilon_{\alpha\beta\gamma} t_{\gamma\lambda i} \sum_j \frac{m_j}{\rho_j} (x_{\beta i} - x_{\beta j}) (W_{,\lambda})_{ij} + O(h^2). \quad (36)$$

The sum over j in this expression can be reduced to $\delta_{\gamma\lambda}$ by going over to the integral formulation (thus, particle errors have to be neglected) which leads – due to the symmetry of $t_{\gamma\lambda}$ – to the vanishing of the viscous contribution, too, at least to order h^2 .

Since we have always neglected surface terms in the SPH formulation, this result is in full agreement with the angular momentum conservation as expressed by Eq. (34) for the continuum model.

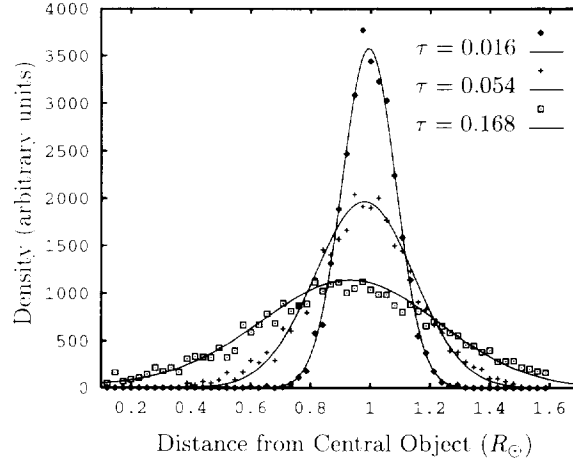


Fig. 1. Comparison of the density in the ring test problem for different times τ . The analytic solution is shown as lines and the results of the SPH simulation as markers.

4. Tests and astrophysical applications

4.1. Numerical tests of the viscosity

We have performed a number of numerical calculations in order to test our newly derived expression Eqs. (18)–(20) for the physical viscosity. First we simulated the two-dimensional flow of a fluid under the influence of a constant gravitational force between two parallel plates with the appropriate boundary conditions. We obtained a stationary flow with the correct quadratic velocity profile.

As a second test we have calculated the development of a thin gas ring on a Keplerian orbit around a point mass including viscosity which is parameterized by $\nu = \eta/\rho = \text{const.}$. The problem has been reduced to two dimensions by integrating all physical quantities over the height of the disk

$$\rho^{(2)} = \int dz \rho^{(3)}, \quad p^{(2)} = \int dz p^{(3)} \quad \text{etc.}, \quad (37)$$

where the superscripts ⁽²⁾ and ⁽³⁾ denote two- and three-dimensional quantities, respectively. The external force F_α from Eq. (11) is in this case simply given by the gradient of the Newtonian potential. This problem can be solved analytically (Pringle 1981, Frank et al. 1992); the two-dimensional time-dependent mass density distribution $\rho^{(2)}(x, \tau)$ reads

$$\rho^{(2)}(x, \tau) = \frac{m}{\pi r_0^2} \frac{1}{\tau} x^{-1/4} \exp\left(-\frac{1+x^2}{\tau}\right) I_{1/4}\left(\frac{2x}{\tau}\right) \quad (38)$$

in terms of the dimensionless variables $x = r/r_0$, $\tau = t/t_{\text{visc}}$ with the viscous time scale $t_{\text{visc}} = r_0^2/12\nu$ (the surface mass density $\rho^{(2)}$ is usually denoted by Σ). This solution corresponds to the initial condition

$$\rho^{(2)}(r, 0) = \frac{m}{2\pi r_0} \delta(r - r_0). \quad (39)$$

The results of the simulation together with the analytic solution can be seen in Fig. 1 where the following parameters have been used: 10 000 particles, $\nu = 7.7 \times 10^8 \text{ m}^2/\text{s}$, and $r_0 = R_\odot = 6.96 \times 10^8 \text{ m}$; the smoothing length was $h = 4.5 \times 10^{-2} r_0$. In order to avoid the singular behaviour of the density at $\tau = 0$ we have started the simulation at $\tau = 0.016$ where the density profile has been sampled according to the solution (38). For the

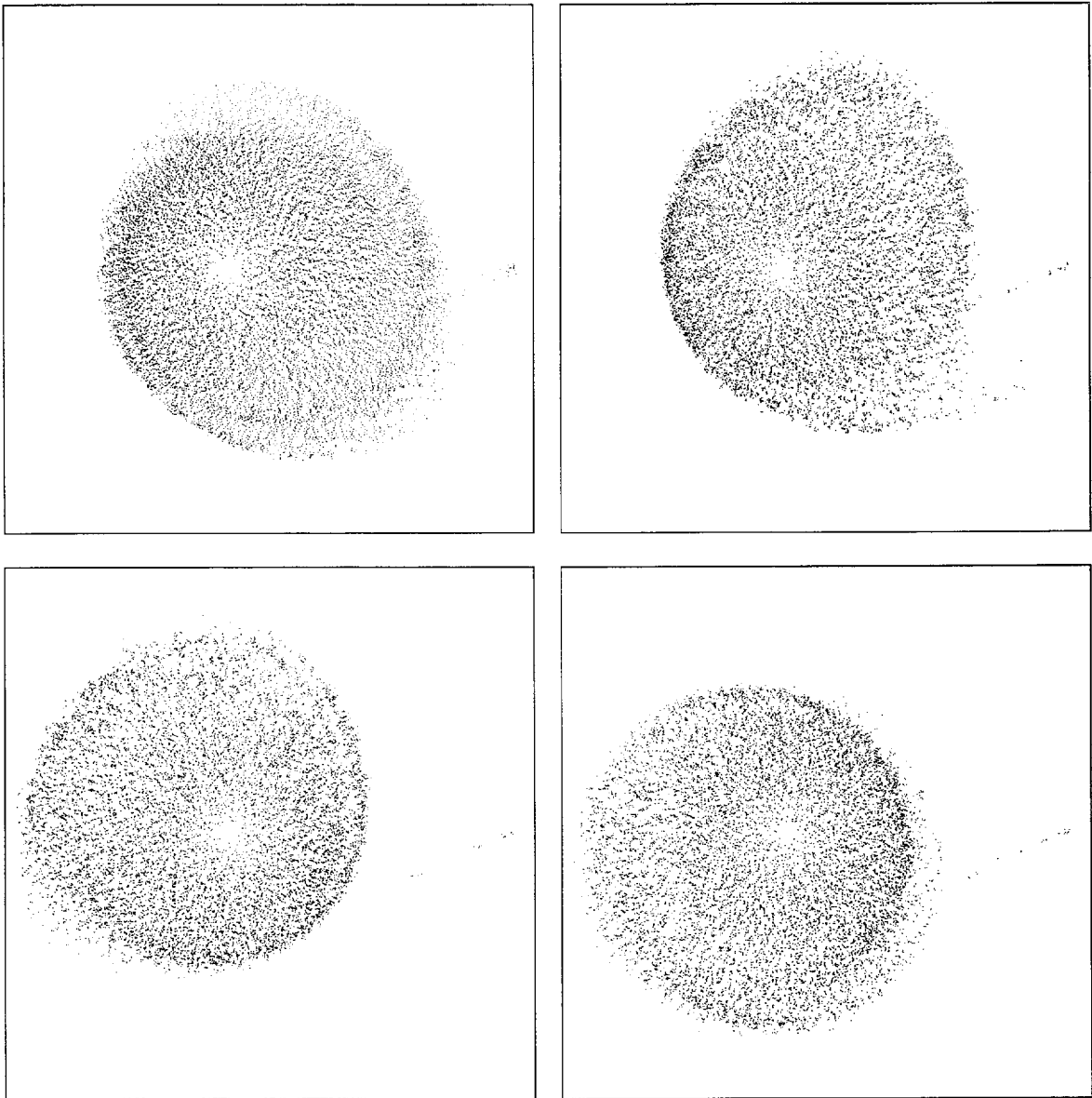


Fig. 2. Simulation of a precessing accretion disk in a binary system.

parameters chosen the radial velocity v_r derived from the solution (38) is less than 5 percent of the orbital velocity v_ϕ . Particles which are closer than $0.05r_0$ and farther away than $2r_0$ from the central object are removed from the simulation. The results of the simulation (see Fig. 1) are in good agreement with the analytic solution. In particular the flow evolves on the correct viscous time scale.

4.2. Simulation of accretion disks

Next we simulate a geometrically thin accretion disk in a binary system. Similar SPH simulations for accretion disks in binary systems have been performed by a number of authors (e.g., Whitehurst 1988, Lubow 1992) in

various contexts. However, in many cases the way in which the viscous forces have been implemented is not quite clear.

Describing the gas dynamics in a reference frame co-rotating with the orbital motion the external force F_α reads

$$F_\alpha = -\Phi_{,\alpha} - (\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}))_\alpha - 2(\boldsymbol{\omega} \times \mathbf{v})_\alpha \quad (40)$$

where $\boldsymbol{\omega}$ is the orbital angular velocity and Φ stands for the gravitational potential of the two point masses. Again the problem is reduced to two dimensions by integrating over the height of the disk.

As a consequence of our two-dimensional model together with the assumption that the cooling processes have a much shorter time scale than the hydrodynamical processes the dissipated energy will be instantly radiated away from both sides of the accretion disk. We define the effective temperature T_{eff} according to

$$2\sigma T_{\text{eff}}^4 = \int dz T \frac{ds}{dt} \rho^{(3)} = T \frac{ds}{dt} \rho^{(2)}. \quad (41)$$

In order to close the set of equations we use the equation of state of an ideal gas with the gas temperature T equal to the effective temperature T_{eff}

$$p^{(2)} = \frac{\rho^{(2)} k_B T}{\mu m_H}. \quad (42)$$

In Fig. 2, a simulation of a precessing accretion disk in a binary system is shown. The mass of the primary M_1 in the center of the disk is $4.5M_\odot$ and the mass of the secondary is $0.5M_\odot$. The orbital period of the system is $P = 0.2d$ and the infalling mass stream has a mass transfer rate of $\dot{M} = 4 \times 10^{-9}M_\odot/a$. The simulation was done with a viscous parameter of $\nu = 1.4 \times 10^{10} \text{m}^2/\text{s} = \text{const}$.

The simulation is done with about 11 000 particles. The diameter of the smoothing kernel is 1/80 of the region shown in the Fig. 2.

Accretion disks in binary systems show precession, if the systems have mass ratios of about $M_1/M_2 = 5 \dots 20$. This is due to unstable simple periodic orbits in the Roche potential (Paczynski 1977). Lubow (1991a,b) has shown how the tidal and hydrodynamic modes can lead to a precessing accretion disk. The precession can explain some features of the superhump phenomena found in SU UMa stars. The period of the precessing disks found in simulations is very close to the period of observed superhumps in the beginning of a superoutburst (Frank et al 1992).

5. Selected topics in code implementation

In this section we give an introduction into some performance relevant techniques which can be used for SPH codes. The discussion is restricted to two dimensions only, but can easily be extended to three dimensions.

5.1. Smoothing kernel and time integration

We use the W_4 spline as proposed by Monaghan (1985) with $q = r/h$:

$$W_4(q) = \begin{cases} \frac{40}{7h^2\pi} (1 - 6q^2 + 6q^3), & q < \frac{1}{2}, \\ \frac{80}{7h^2\pi} (1 - q)^3, & \frac{1}{2} \leq q < 1, \\ 0, & q \geq 1. \end{cases} \quad (43)$$

Since our applications are two-dimensional, the spline in Eq. (31) is normalized for 2D. This intervecating kernel gives accurate results for the density and is used for all intervecations. All simulations presented here are performed with particles of equal mass, although this is not necessary in general.

The largest value of the time step Δt is given by the Courant condition

$$|v\Delta t/s| < 1 \quad (44)$$

(originally derived for finite difference methods). The grid resolution s can be replaced by the smoothing parameter h . Therefore we tried to find an integrator which permits the largest allowed time steps without losing accuracy. Because the right hand side of our equation of motion is quite complicated to evaluate, we have been looking for an integrator which needs the least evaluations of the right hand side.

In order to integrate the equations of motion we have chosen the Runge-Kutta-Fehlberg RKF5(4)FM method (Dormand & Prince 1980), a high precision combined fifth and fourth order RKF method with adaptive step size. We compared this method with a standard Adams-Bashforth predictor-corrector method. It turned out that the RKF algorithm gives better accuracy for a given number of force evaluations.

5.2. Implementation of the interactions

The most time consuming part of particle codes is to identify all interacting particle pairs. In order to reduce the computational effort the particle positions are determined with respect to some regular grid where one can get the neighbouring particles very fast (Sedgewick 1992). The details of such an algorithm are described in the following. There are other special algorithms to perform this task by clustering the particles while simultaneously calculating gravitational forces (Benz 1990, Hernquist and Katz 1989).

The SPH simulation needs many evaluations of terms of the form ‘sum over all particle interactions’. A typical example for such a sum is the acceleration by the pressure gradient

$$\frac{dv_{\alpha i}}{dt} = -m \sum_{j \neq i} \frac{p_j + p_i}{\rho_i \rho_j} (W_{,\alpha})_{ij}. \quad (45)$$

For efficiency reasons one should determine all particle-particle pairs with a distance of less than the smoothing length h only once and store them in a linear list. Moreover, it is best to store also the derivative of the spline at the same time. The evaluation of a spline needs a conditional branch which is ineffective on most computer architectures.

5.2.1. Scalar implementation

Using the antisymmetry of the forces is one of the optimizing tricks for scalar architectures. One can use this property and calculate and store only one half of all interactions. So the evaluation of expressions like Eq. (45) can be performed by a single loop over all interactions (n_{ww} is the number of interaction pairs):

```
do m=1,nww
  i = listi(m)
  j = listj(m)
  temp = mass*(p(j)+p(i))/(rho(i)*rho(j))
  tmp = temp*listwsx(m)
  dvx(i) = dvx(i) - tmp
  dvx(j) = dvx(j) + tmp
  tmp = temp*listwsy(m)
  dvy(i) = dvy(i) - tmp
  dvy(j) = dvy(j) + tmp
enddo
```

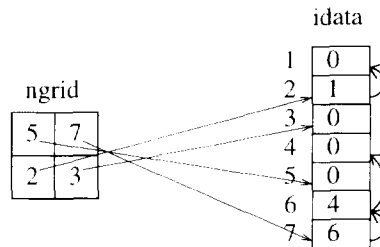


Fig. 3. The linked lists `ngrid` and `idata` for an example configuration of 7 particles. The upper left cell contains particle number 5, the upper right cell contains particles number 7, 6 and 4. The lower two cells contain the particles with numbers 2, 1 and 3, respectively.

The lists `listi` and `listj` contain the indices of the particles in the interaction pair m . The two spatial derivatives of the spline are calculated once and have been stored in the arrays `listwsx` and `listwsy`, respectively.

For building the list of interactions it is necessary to use an efficient technique. In general only a small fraction of all possible particle pairs is close enough to each other to contribute to the sum, so testing all particle-particle distances is very inefficient. It is common to use an auxiliary spatial grid to sort the particles into cells and to restrict the search to the neighbouring cells. In order to minimize the memory requirement one uses linked lists.

If the number of particles is `NPART`, the algorithm uses an array `idata(NPART)` for the linked lists and a two-dimensional array `ngrid(NCELLX, NCELLY)` for the indices of each linked list. The `ngrid` array should be initialized to 0, indicating that no particles are in the cells. Particle i is included in the linked list by assigning the value of the cell of the current position to `idata(i)`. Afterwards the cell in `ngrid` is updated in order to point to the current node in the linked list `idata`.

With this algorithm linked lists are constructed in `idata` with the head pointer in the array `ngrid`. The trick is that the links in `ngrid` and `idata` are used both as a reference to the elements in the vector `idata` and as an index of the particle as shown in Fig. 3. The following code fragment shows how to build the linked lists.

```
do i=1, NPART
  ix=int(posgrid(posx(i)))
  iy=int(posgrid(posy(i)))
  idata(i) = ngrid(ix,iy)
  ngrid(ix,iy) = i
enddo
```

The function `posgrid` calculates the position inside the auxiliary grid from the real particle positions `posx` or `posy`. One can run through the linked list starting in the cell $j = \text{ngrid}(ix, iy)$ by dereferencing the linked list $j = \text{idata}(j)$ until the end $j = 0$ is found. All possible interactions for a particle can be found by searching in the linked lists of the neighbouring cells of the particle i . Because the list `idata` is sorted in descending order by construction, one can stop when an index is found which is lower than i . The result is stored in linear lists `listi` and `listj` (hx and hy denote the smoothing length h in cell units):

```
nww = 0
do i=1, NPART-1
  xr = posgrid(posx(i))
  yr = posgrid(posy(i))
  minx = max(int(xr - hx), 1) ! minimum for x, minx > 0
  maxx = min(int(xr + hx), NCELLX) ! max
  miny = max(int(yr - hy), 1) ! minimum for y, miny > 0
```

```

maxy = min(int(yr + hy),NCELLY) ! max

do ix=minx,maxx
  do iy=miny,maxy
    j=ngrid(ix,iy)
1    continue
    if ( j.gt.i ) then
      dx = posx(i) - posx(j) ! calculate distance
      dy = posy(i) - posy(j)
      d = dx*dx + dy*dy
      if (d.le.h*h) then ! distance < h
        nww = nww+1
        listi(nww) = i
        listj(nww) = j
      endif
      j = idata(j)
      goto 1
    endif
  enddo ! iy
enddo ! ix
enddo ! i

```

5.2.2. Vector implementation

In this section we want to show how to get higher performance with the help of a vector computer using a vectorizing code.

The main ideas are the following: One cannot use the symmetry in the forces, because this would cause data recursion which makes vectorization impossible. So one has to calculate the interactions twice. The loops over all interactions have to be split into several independent loops. For instance, one can split a loop over 10 000 particles with a maximum of 10 interactions per particle into 10 vectorizing loops of independent interactions. One has to inform the compiler that the inner loop is a loop of a permutation and therefore does not contain any recursion. The vectorizing code for calculating the pressure gradient is shown below:

```

do k=1,nloops
C$DIR no_recurrence
  do m=mstart(k),mstart(k+1)-1
    i = loopsi(m)
    j = loopsj(m)
    tmp = mass*(p(j)+p(i))/(rho(i)*rho(j))
    dvx(i) = dvx(i) - tmp*listwsx(m)
    dvy(i) = dvy(i) - tmp*listwsy(m)
  enddo
enddo

```

The inner loop over m can be vectorized, i.e. the block of interaction indices m starts at $mstart(k)$ and ends at $mstart(k+1)-1$ and contains an index only once. With the compiler directive `C$DIR no_recurrence` this information is passed to the compiler, which can now generate a vector loop over m . This directive is specific to CONVEX compilers. CRAY compilers need the directive `C$DIR ivdep`. Here, the lists `loopsi` and `loopsj` contain the particle indices of the interactions sorted into distinct loops as shown in Fig. 4. How to construct

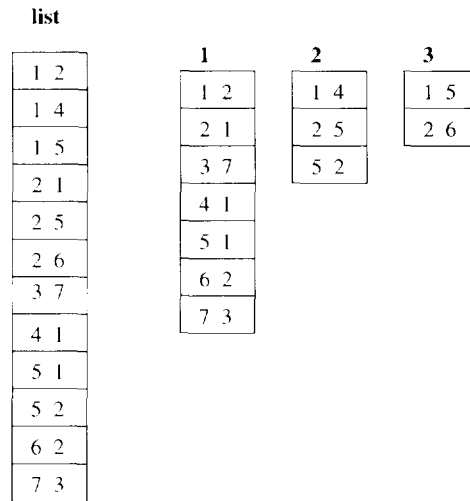


Fig. 4. Example for splitting a list *list* of interactions into several loops 1, 2 and 3 in order to avoid data recursion. The *n*th loop consists of all *n*th interactions.

these lists from the original interaction list is described in the following.

The interactions can be split into independent parts with the help of the observation that the algorithm for finding all possible particle-particle interactions presented above generates the interactions in ascending order. Thus it can be used to distribute all interactions of particle *i* into several independent loops.

First we have to find the boundaries of the clusters in the first index: a vector loop which scans the list for differences is sufficient. After running this loop the array *newp* contains links to the boundaries of clusters of interactions.

```

newp(1) = 1
ic = 1
do m=2,nww
  if (listi(m).ne.listi(m-1)) then
    ic = ic + 1
    newp(ic) = m
  endif
enddo
ic = ic + 1
newp(ic) = nww+1
icmax = ic

```

Secondly, we scatter the contents of the interaction list into a large array *mtab* where they can be read out in order to create a new set of indices. All loops vectorize, but the loop over *m* performs badly, because under normal conditions the loop count is only in the order of the number of interactions per particle. One can enforce scalar code for this special loop to get a better performing code.

```

! prepare sorting indices
do k=1,MAXNLOOPS
  mcount(k) = (k-1)*NPART
enddo

```

```

! scatter the different interactions
do ic=1,icmax-1
C$DIR no_vector
  do m=1,newp(ic+1)-newp(ic)
    mcount(m) = mcount(m) + 1
    mtab(mcount(m)) = newp(ic)+m-1
  enddo
enddo

```

At last we gather the indices back to an array, calculating the spline at the same time.

```

! determine the number of nonempty loops
nloops = 0
do k=1,MAXNLOOPS
  if ( mcount(k).ne.(k-1)*NPART ) nloops = k
enddo

! build start list
m = 1
do k=1,nloops
  mstart(k) = m
  m = m + mcount(k) - (k-1)*NPART
enddo
mstart(nloops+1) = m

! build loops list
m0 = 0
do k=1,nloops
C$DIR no_recurrence
  do m=1,mstart(k+1)-mstart(k)
    loopsi(m0+m) = listi(mtab((k-1)*NPART+m))
    loopsj(m0+m) = listj(mtab((k-1)*NPART+m))
  enddo
  m0 = m0 + mstart(k+1)-mstart(k)
enddo

```

c calculate spline and density rho

```

.
.
.
enddo
m0 = m0 + mstart(k+1)-mstart(k)
enddo

```

With the help of these tricks one can get a better vector performance for SPH codes. But the total performance gain, compared to the scalar code, is limited because the symmetry in the particle forces cannot be used and one is forced to calculate the interactions twice.

Since the recalculation of all particle-particle interactions is one of the most expensive subroutines, it should be avoided. This can be achieved by taking profit of the requirement that the particle displacement within one time step should be less than one smoothing length of the particle itself (cf. Eq. (44)). This fact can be used to calculate the possible interactions for a larger extent and to employ the list of interactions several times on successive time steps. We used this trick in combination with the RKF integrator. Since this integration scheme

Table 1

Execution times in seconds for a benchmarking problem (1/500 of the orbital period of a binary system with about 7500 particles in the accretion disk) on several machines

Machine	Code	
	Scalar	Vector
CONVEX 3860	32.3	9.4
SGI Indigo R3000	52.2	103.0
486DX66 (Linux)	127.2	–

needs 7 evaluations of the forces – where the last one can be used again – it is sufficient to determine all the interactions for a radius which is twice as large as the actual smooting length. This list can be used for every evaluation within one integration step. Moreover, it is now trivial to use the interactions repeatedly if the integration fails because the time step was too large.

In order to compare the different codes on different machines we made some benchmarks which are shown in Table 1. The timings in the ‘scalar code’ column are for the scalar code with automatic vectorization turned on for the CONVEX. The column ‘vector code’ shows timings of the vector code on two typical machines. Because of memory limitations it does not run well on the Linux PC. The CONVEX results show that the vector code is effective. The doubling of the interaction calculation in the vector code obviously is responsible for the results on the SGI machine. Because the SPH application uses a lot of integer operations the performance of the PC is rather high.

References

- Benz W. 1990, in: *The Numerical Modelling of Nonlinear Stellar Pulsations, Problems and Prospects*, ed. J.R. Buchler (Kluwer Academic Publishers, Dordrecht).
- Dormand J.R., Prince P.J., 1980, *J. Comput. App. Math.* **6**, 19.
- Flebbe O., Münzel S., Herold H., Riffert H., Ruder H., 1994, *Astrophys. J.*, in press.
- Frank J., King A.R., Raine D.J., 1992, *Accretion Power in Astrophysics*, 2nd ed. (Cambridge University Press).
- Gingold, R.A., Monaghan J.J., 1977, *Mon. Not. Roy. Astron. Soc.* **181**, 375.
- Harlow F.H., 1964, *Meth. Comput. Phys.* **3**, 319.
- Hernquist L., Katz N., 1989, *Astrophys. J. Suppl. Ser.* **70**, 419.
- Laguna P., 1994, Preprint PSU-ASTRO 94/2-1, submitted to *Astrophys. J.*
- Lucy L.B., 1977, *Astronom. J.* **82**, 1013.
- Landau L.D., Lifschitz E.M., 1991, *Hydrodynamik* (Akademie-Verlag Berlin).
- Lubow S.H., 1991a, *Astrophys. J.* **381**, 259.
- Lubow S.H., 1991b, *Astrophys. J.* **381**, 268.
- Lubow S.H., 1992, *Astrophys. J.* **401**, 317.
- Martin T.J., Pearce F.R., Thomas P.A., 1993, Preprint SUSSEX-AST-MPT-1, Sussex University.
- Monaghan J.J., 1985, *Comput. Phys. Rep.* **3**, 71.
- Monaghan J.J., 1992, *Ann. Rev. Astron. Astrophys.* **30**, 543.
- Monaghan J.J., Gingold R.A., 1983, *J. Comput. Phys.* **52**, 374.
- Paczynski B., 1977, *Astrophys. J.* **216**, 822.
- Pringle J.E., 1981, *Ann. Rev. Astron. Astrophys.* **19**, 137.
- Sedgewick, 1992, *Algorithmen*, 2nd ed. (Addison-Wesley, Reading, MA).
- Whitehurst R., 1988, *Mon. Not. Roy. Astron. Soc.* **232**, 35.