Matrix Based SPH Formulation

Štěpán Müller

Czech Technical University in Prague, Faculty of Mechanical Engineering

May 2024

1 Introduction

In this article, a matrix-based formulation of a numerical SPH scheme is presented. For demonstration, the following weakly compressible SPH scheme in 2 dimensions will be used:

$$\frac{D\rho_i}{Dt} = -\rho_i \sum_j (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_{ij} V_j + c_0 \sum_j (\rho_j - \rho_i) \frac{(\mathbf{r}_j - \mathbf{r}_i) \cdot \nabla W_{ij}}{|| (\mathbf{r}_j - \mathbf{r}_i) ||^2} V_j$$

$$\frac{D\mathbf{u}_i}{Dt} = -\frac{1}{\rho_i} \sum_j (p_j + p_i) \cdot \nabla W_{ij} V_j + \mathbf{f}_i + \nu \frac{\rho_0}{\rho_i} \sum_j \pi_{ij} \nabla W_{ij} V_j$$

$$\frac{D\mathbf{r}_i}{Dt} = \mathbf{u}_i$$

$$p_i = c_0^2 (\rho_i - \rho_0)$$

where W_{ij} is the Gaussian Kernel, resulting in

$$\nabla W_{ij} = \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \cdot \frac{2}{\pi h^4} \cdot exp\left(-\frac{(x_j - x_i)^2 + (y_j - y_i)^2}{h^2} \right)$$

and π_{ij} equals to

$$\pi_{ij} = 8 \cdot \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|| \mathbf{r}_j - \mathbf{r}_i ||^2}$$

A common algorithmization approach is to loop for i over all particles and calculate their interactions with appropriate j particles. The aim of the matrix based method presented in this article is to greatly reduce the amount of operations performed in a loop and replace them with matrix and vector operations.

2 Constructing the matrix based formulation

2.1 Interaction matrix I

The matrix method starts by identifying possible interacting particle pairs (for example using the bin sort method) and storing this information in the interaction matrix I. I is a sparse square symetrical binary matrix of size $N \times N$, where N is the number of particles.

$$I = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Each row i represents the interactions of particle i with particles j, where j is the column. Here the first row would mean that particle 1 does not interact with itself, it interacts with particle 2 and does not interact with particle 3.

I can be stored for example in the COO sparse format. Finding the matrix I is the only loop necessary to algorithmize the process.

2.2 Vectors of variables

The particle variables will be referred to as 1D vectors of length N, for example vector

$$\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

stores the x coordinate of $N = 3$ particles.

2.3 Matrices of differences

Matrices of differences will replace all members that involve $j - i$ subtraction, such as $x_j - x_i$. At first a matrix Mx is defined, which results from a column-wise multiplication of $\vec{x}$ and I, such as

$$Mx = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & x_2 & 0 \\ x_1 & 0 & x_3 \\ 0 & x_2 & 0 \end{bmatrix}$$

The matrix of differences Dx is defined:

$$Dx = Mx - Mx^T = \begin{bmatrix} 0 & x_2 & 0 \\ x_1 & 0 & x_3 \\ 0 & x_2 & 0 \end{bmatrix} - \begin{bmatrix} 0 & x_1 & 0 \\ x_2 & 0 & x_2 \\ 0 & x_3 & 0 \end{bmatrix} = \begin{bmatrix} 0 & x_2 - x_1 & 0 \\ x_1 - x_2 & 0 & x_3 - x_2 \\ 0 & x_2 - x_3 & 0 \end{bmatrix}$$

Using the same approach, difference matrices $Dy, Du, Dv, D\rho$ can be calculated, with the exception of difference matrix Dp, which uses addition instead of subtraction due to the chosen SPH scheme.

It requires emphasizing that all involved matrix by vector multiplications should utilize the sparsity of the matrix to prevent $O(N^2)$ time complexity, which would make all effort for further optimization entirely pointless.

2.4 Matrix ∇W_x and ∇W_y

Hereafter, a simplified notation will be used:

$$x_j - x_i = dx_{ij}$$

Steps to construct the ∇W matrices follow:

$$Mr^2 = Dx^2 + Dy^2 = \begin{bmatrix} 0 & x_{12}^2 & 0 \\ x_{21}^2 & 0 & x_{23}^2 \\ 0 & x_{32}^2 & 0 \end{bmatrix} + \begin{bmatrix} 0 & y_{12}^2 & 0 \\ y_{21}^2 & 0 & y_{23}^2 \\ 0 & y_{32}^2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & r_{12}^2 & 0 \\ r_{21}^2 & 0 & r_{23}^2 \\ 0 & r_{32}^2 & 0 \end{bmatrix}$$

Using the exponential function, only non-zero elements are taken into account:

$$\nabla W_{size} = exp_{arg \neq 0}\left(\frac{-1}{h^2} Mr^2\right) \cdot \frac{2}{\pi h^4}$$

Eventually, matrices ∇W_x and ∇W_y can be acquired using element-wise multiplication:

$$\nabla W_x = Dx \nabla W_{size} = \begin{bmatrix} 0 & \frac{2dx_{12}}{\pi h^4} \cdot exp\left(-\frac{dx_{12}^2 + dy_{12}^2}{h^2}\right) & 0 \\ \dots & 0 & \dots \\ 0 & \dots & 0 \end{bmatrix}$$

$$\nabla W_y = Dy \nabla W_{size} = \begin{bmatrix} 0 & \frac{2dy_{12}}{\pi h^4} \cdot exp\left(-\frac{dx_{12}^2 + dy_{12}^2}{h^2}\right) & 0 \\ \dots & 0 & \dots \\ 0 & \dots & 0 \end{bmatrix}$$

2.5 Matrix π

Similarly to construction of ∇W, a matrix π can be constructed using element-wise multiplications:

$$\pi = 8 \cdot (DuDx + DvDy) \cdot \frac{1}{Mr^2}$$

3 Final matrix SPH scheme

For the chosen SPH scheme, the matrix formulation gives:

$$\frac{\overrightarrow{D\rho}}{Dt} = -\vec{\rho}\left[(Du\nabla W_x + Dv\nabla W_y)\cdot\vec{V}\right] + c_0\left[\frac{D\rho\,(Dx\nabla W_x + Dy\nabla W_y)}{\sqrt{Mr^2}}\right]\cdot\vec{V}$$
$$\frac{\overrightarrow{Du}}{Dt} = \frac{-1}{\vec{\rho}}\left[(Dp\nabla W_x)\cdot\vec{V}\right] + \frac{\nu\rho_0}{\vec{\rho}}\left[(\pi\nabla W_x)\cdot\vec{V}\right] + \vec{g_x}$$
$$\frac{\overrightarrow{Dv}}{Dt} = \frac{-1}{\vec{\rho}}\left[(Dp\nabla W_y)\cdot\vec{V}\right] + \frac{\nu\rho_0}{\vec{\rho}}\left[(\pi\nabla W_y)\cdot\vec{V}\right] + \vec{g_y}$$
$$\frac{\overrightarrow{Dx}}{Dt} = \vec{u}$$
$$\frac{\overrightarrow{Dy}}{Dt} = \vec{v}$$
$$\vec{p} = c_0^2\,(\vec{\rho} - \vec{\rho_0})$$

where the $\cdot$ symbol indicates basic matrix by vector multiplication, whereas no dot suggests element-wise multiplication.

4 Data handling optimization

4.1 Data vector format

Because matrices

$$Dx, Dy, Du, Dv, Dp, Mr^2, \nabla W_{size}, \nabla W_x, \nabla W_y, \pi$$

all have the same structure, sharing the row and column indices of I, most operations can be only performed on the data vector using the COO matrix format. Using the example notation

$$\vec{Dx} = Dx.data$$

the scheme can be rewritten to a more efficient vector format. The only need to reuse the full matrix format comes from the $\cdot$ multiplication by vector of volumes $\vec{V}$:

$$\frac{\overrightarrow{D\rho}}{Dt} = -\vec{\rho}\left[to\_coo\_matrix\left(\vec{Du}\nabla\vec{W}_x + \vec{Dv}\nabla\vec{W}_y\right)\cdot\vec{V}\right] + to\_coo\_matrix\left[\frac{c_0\vec{D}\rho\left(\vec{Dx}\nabla\vec{W}_x + \vec{Dy}\nabla\vec{W}_y\right)}{\sqrt{\vec{Mr}^2}}\right]\cdot\vec{V}$$
$$\frac{\overrightarrow{Du}}{Dt} = \frac{1}{\vec{\rho}}\left[to\_coo\_matrix\left(-\vec{D}p\nabla\vec{W}_x + \nu\rho_0\vec{\pi}\nabla\vec{W}_x\right)\cdot\vec{V}\right] + \vec{g_x}$$
$$\frac{\overrightarrow{Dv}}{Dt} = \frac{1}{\vec{\rho}}\left[to\_coo\_matrix\left(-\vec{D}p\nabla\vec{W}_y + \nu\rho_0\vec{\pi}\nabla\vec{W}_y\right)\cdot\vec{V}\right] + \vec{g_y}$$

4.2 Simplified way to calculate data vectors

Furthermore, there is a simplier way to calculate the difference data vectors $\vec{Dx}, \vec{Dy}, \vec{Du}, \vec{Dv}, \vec{D\rho}, \vec{Dp}$ by only using row and column vector of the sparse COO matrix I, instead of using full sparse multiplication:

$$\vec{Dx} = \vec{x}[columns] - \vec{y}[rows]$$

Where $\vec{x}[columns]$ means selecting x_i so that i goes through all column indices of I. Using I from previous chapters as example:

$$I = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow I.rows = \begin{bmatrix} 1 & 2 & 2 & 3 \end{bmatrix}; I.columns = \begin{bmatrix} 2 & 1 & 3 & 2 \end{bmatrix}; I.data = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$
$$\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}; \vec{x}[columns] = \begin{bmatrix} x_2 & x_1 & x_3 & x_2 \end{bmatrix}; \vec{x}[rows] = \begin{bmatrix} x_1 & x_2 & x_2 & x_3 \end{bmatrix}$$
$$\vec{Dx} = \begin{bmatrix} x_2 - x_1 & x_1 - x_2 & x_3 - x_2 & x_2 - x_3 \end{bmatrix}$$

As shown, this method provides the same result for $\vec{Dx}$ as the method from 2.3.

5 Procedural modification of the interaction matrix

In the previous chapters, it was shown how a full SPH calculation scheme can be built based on a single interaction matrix I. Every iteration it is firstly required to build I, as I might be different every iteration due to the movement of interacting particles.

The most basic way to build I is to use the bin sort algorithm and construct I from scratch every iteration. This suggests to loop over the bins, which might not be performed as efficiently as the rest of the matrix-based algorithm.

Therefore in this chapter, a different method of managing I is presented.

5.1 Standard bin sort method

At first, let's look at the standard bin sort method of generating I for a 2 dimensional case.

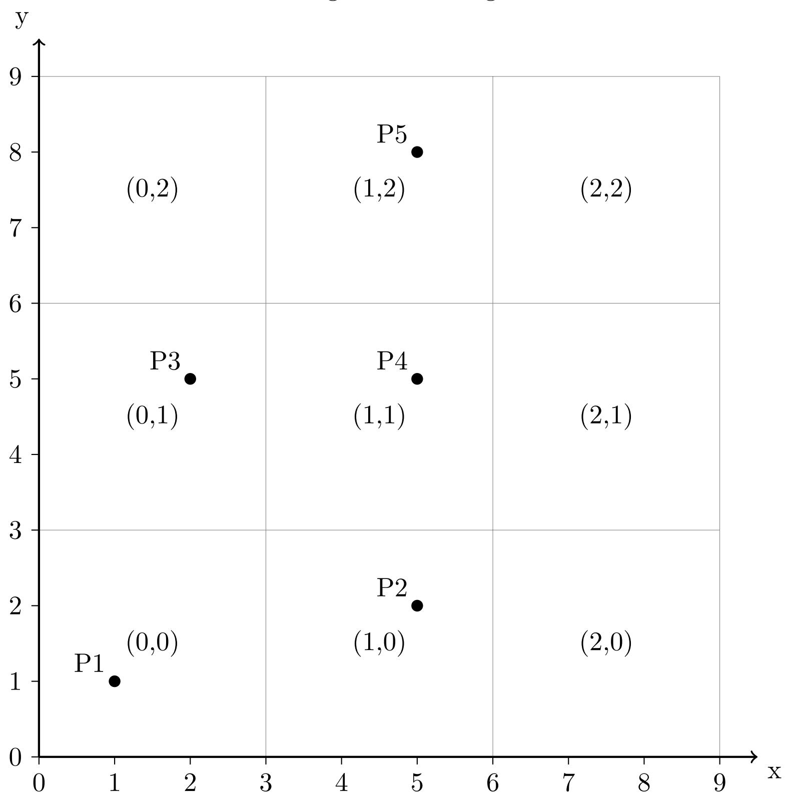
The bin sort method divides the XY coordinate space into bins and assigns a corresponding bin to every particle (or assigns a set of particles to every bin).

Let's consider 5 particles:

$$\vec{x} = \begin{bmatrix} 1 & 5 & 2 & 5 & 5 \end{bmatrix}$$

$$\vec{y} = \begin{bmatrix} 1 & 2 & 5 & 5 & 8 \end{bmatrix}$$

Let's sort them into 9 bins using the following labels:



The information about where each point belongs can be represented as:

$$\vec{binx} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$\vec{biny} = \begin{bmatrix} 1 & 2 & 1 & 1 & 2 \end{bmatrix}$$

Looping over all bins and browsing adjacent bins for interacting particles gives I:

$$I = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

5.2 Prerequisites of the procedural modification method

The following method of generating I is based on the prerequisite, that I usually changes only very slightly between each iteration.

Let there be N particles distributed approximately evenly with an average spacing s. The weighting length h can be chosen as for example $1.5S$. Consequently, when using the Gaussian kernel, the size of one bin is $3S = 2h$. The simulation runs with a constant timestep T and particles have an average velocity of u.

Under these circumstances, the average number of particles that leave their bin during one iteration will be on the order of

$$N_{bin\ changes} \approx N \cdot \frac{u \cdot T}{3S} = N \cdot \frac{u \cdot T}{2h}$$

The timestep T can be derived from the CFL condition (missing reference):

$$T = \alpha \cdot \frac{2h}{c}$$

where c is velocity of sound and α can be set constant to 0.4 according to (reference). Substituting for T in the first equation:

$$N_{bin\ changes} \approx N \cdot \frac{u \cdot 0.4 \cdot \frac{2h}{c}}{2h} = N \cdot \frac{0.4u}{c} = N \cdot 0.4Ma$$

For water flowing at $10m/s$:

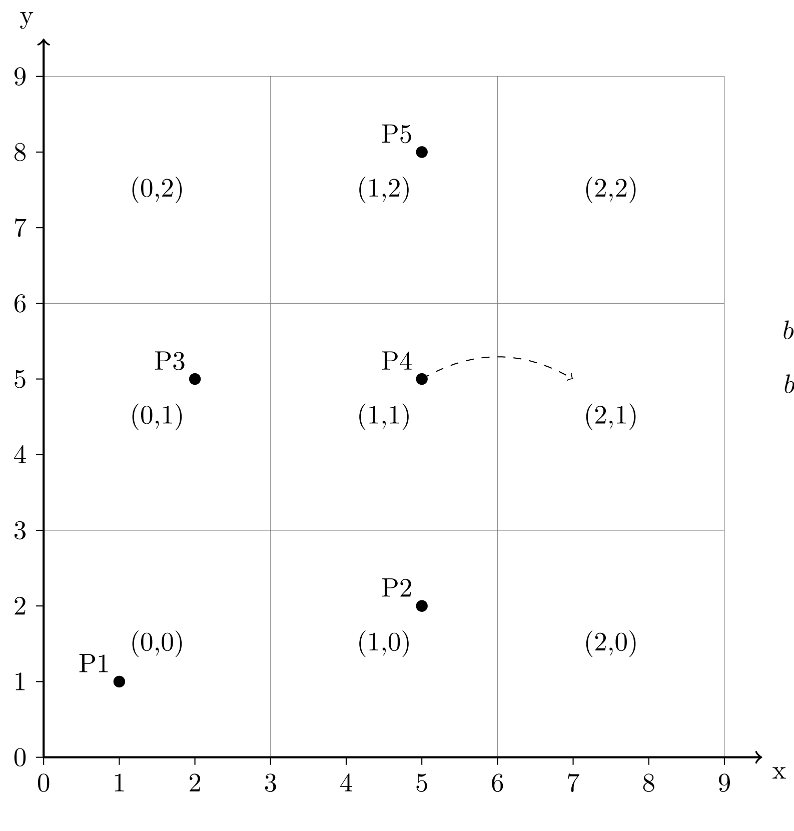
$$N_{bin\ changes} \approx N \cdot \frac{0.4 \cdot 10}{1500} \approx 3 \cdot 10^{-3}N$$

As shown, the number of bin changes per iteration will generally be much lower than number of particles. The goal of the following method is to utilize this and avoid generating brand new I every iteration.

5.3 Procedural modification method

Let's assume that the I_{k-1} was created in the previous iteration $k-1$. It is also known which bins were occupied in the previous iteration:

$$Iteration\ k-1:$$



$$\vec{x}_{k-1} = \begin{bmatrix} 1 & 5 & 2 & 5 & 5 \end{bmatrix}$$

$$\vec{y}_{k-1} = \begin{bmatrix} 1 & 2 & 5 & 5 & 8 \end{bmatrix}$$

$$\vec{binx}_{k-1} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$\vec{biny}_{k-1} = \begin{bmatrix} 1 & 2 & 1 & 1 & 2 \end{bmatrix}$$

$$I_{k-1} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

In the next iteration k, the particle $P4$ moves from [5 5] to [7 5], which shifts it to a different bin. The new bin vectors can be calculated by dividing coordinate vectors by bin size $2h = 3$ and ceiling them to the closest lower integer:

$$\vec{x_k} = \begin{bmatrix}1 & 5 & 2 & 7 & 5\end{bmatrix}$$
$$\vec{y_k} = \begin{bmatrix}1 & 2 & 5 & 5 & 8\end{bmatrix}$$
$$\vec{binx_k} = ceil\left(\frac{\vec{x_k}}{2h}\right) = \begin{bmatrix}0 & 1 & 0 & 2 & 1\end{bmatrix}$$
$$\vec{biny_k} = ceil\left(\frac{\vec{y_k}}{2h}\right) = \begin{bmatrix}1 & 2 & 1 & 1 & 2\end{bmatrix}$$

The new and old bin vectors can be subtracted:

$$\vec{Dbinx} = \vec{binx_k} - \vec{binx_{k-1}} = \begin{bmatrix}0 & 1 & 0 & 2 & 1\end{bmatrix} - \begin{bmatrix}0 & 1 & 0 & 1 & 1\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}$$
$$\vec{Dbiny} = \vec{biny_k} - \vec{biny_{k-1}} = \begin{bmatrix}1 & 2 & 1 & 1 & 2\end{bmatrix} - \begin{bmatrix}1 & 2 & 1 & 1 & 2\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Absolute values of $\vec{Dbinx}$ and $\vec{Dbiny}$ can be added to identify which particles left their bin whichever direction. The resulting sparse vector should be clipped to 1 from the top to allow further operations:

$$\vec{Dbin} = clip(\mid \vec{Dbinx} \mid + \mid \vec{Dbiny} \mid) = \begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}$$

Now, all interactions of particle 4 need to be removed from I_{k-1} to get $I_{cleaned}$ which no longer contains any outdated interactions. This can be done by subtracting two matrices from I, which result from column and row multiplication of I by $\vec{Dbin}$:

$$I_{cleaned} = I_{k-1} - \vec{Dbin}I - \vec{Dbin}^T I =$$

$$= \begin{bmatrix}0 & 1 & 1 & 1 & 0\\ 1 & 0 & 1 & 1 & 0\\ 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 1 & 0\end{bmatrix} - \begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix} \begin{bmatrix}0 & 1 & 1 & 1 & 0\\ 1 & 0 & 1 & 1 & 0\\ 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 1 & 0\end{bmatrix} - \begin{bmatrix}0\\ 0\\ 0\\ 1\\ 0\end{bmatrix} \begin{bmatrix}0 & 1 & 1 & 1 & 0\\ 1 & 0 & 1 & 1 & 0\\ 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 1 & 0\end{bmatrix} =$$

$$= \begin{bmatrix}0 & 1 & 1 & 1 & 0\\ 1 & 0 & 1 & 1 & 0\\ 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 1 & 0\end{bmatrix} - \begin{bmatrix}0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\end{bmatrix} - \begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 0 & 1\\ 0 & 0 & 0 & 0 & 0\end{bmatrix} =$$

$$= \begin{bmatrix}0 & 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\end{bmatrix}$$

All interactions of particle 4 are now cleared up. Next step is to replace them with new interactions.

6 Note about performance

Compared to the standard looping method, which only requires to store vectors as long as N, the matrix based method requires to write vectors of length equal to number of particle pairs. Length of those vectors might range on the order of $10N$ to $100N$.

Despite the memory downside, removing most of the iterative loop using the matrix formulation could provide advantages in cases where memory is not the limit, and computational speed has the highest priority. Implementing the matrix method using efficient multiprocessing for matrix/vector operations needs to be investigated in order to evaluate practical benefits.

7 Conclusion

A matrix based formulation of a chosen SPH scheme was discussed. The principle of the matrix method was explained. A final matrix SPH scheme was derived. Two ways for optimization of the algorithm were shown. Expected performance of the method was briefly discussed. More research and practical implementation is expected to be necessary for proper evaluation.