Computational Astrophysics

Lecture 1: Introduction to numerical methods
Lecture 2: The SPH formulation
Lecture 3: Construction of SPH smoothing functions
Lecture 4: SPH for general dynamic flow
Lecture 5: N-body techniques
Lecture 6: Numerical Implementation

Project Work (5-6 weeks)

David Hobbs

Lund Observatory
ASTM17
Introduction

The SPH formulation in the form of the Navier-Stokes equations was covered in previous lectures.

We have ordinary differential equations (ODE's) with respect to time which can be numerically integrated.

A number of implementation issues are important for realistic simulations

- Variable smoothing length
- Symmetrization of particle interactions
- Zero-energy mode
- Solid boundary treatment
- Pairwise particle interactions
Variable smoothing length

The smoothing length $h$ has a direct impact on the accuracy of the solution and the size of the support domain $\kappa h$.

We should have about 5, 21 or 57 particles in the support domain in 1-, 2- and 3-D respectively.

The smoothing length may need to be adapted in both space and time so $h$ can evolve dynamically.

A simple approach is to update the smoothing length, in d-dimensions, according to the average density as

$$ h = h_0 \left( \frac{\rho_0}{\rho} \right)^{1/d} $$
Symmetrization of particle interactions

If each particle has its own smoothing length, \( h_i \) is not equal to \( h_j \).

Therefore particle \( i \) could exert a force on particle \( j \) but not vise versa, which would violate Newton's third law.

To overcome this one can use averages or geometric means of smoothing lengths for pairs of interacting particles.

Two examples are:

\[
\begin{align*}
    h_{ij} &= \frac{h_i + h_j}{2} \\
    h_{ij} &= \frac{2h_i h_j}{h_i + h_j}
\end{align*}
\]
Zero-energy mode

When evaluating derivatives for **regular** particle distributions the derivative at the particle can be zero and the derivative between neighboring particles can be anti-symmetric. This can result in no strain and is generally undesirable. This can be solved by staggering the particles, solving for velocity at one point and stress at another.

This problem is not so great in SPH as we generally have **irregular** particle distributions and it never arises.

It could be a problem in, for example, simulating atoms on a regular crystalline lattice.
Solid Boundaries

For particles near or on the boundary only particles inside the boundary contribute to the integral.

This is known as particle deficiency.

This gives incorrect solutions.

To solve this problem ghost or virtual particles can be introduced at the boundary to prevent unphysical effects.

**Type I:** On boundary, contribute to kernel and particle approximations for real particles.

**Type II:** Outside boundary, exert repulsive force to prevent interior particles from penetrating.
Nearest neighbors

In SPH, the smoothing function has compact support so only nearest neighboring particles (NNP) interact.

To find them NNP searching algorithms must be used.

Three common types include:

- All-pair search - compare each interaction length with $\kappa h$ to find neighbors - simple but inefficient
- Linked list algorithm - maintain a one off list of neighbors - efficient but not dynamic and requires a large list
- Tree search algorithm - use a subdividing tree search algorithm - efficient and dynamic but more complex

Pairwise interaction can be used to avoid having to use double loops when, for example, calculating summation density.

When searching for nearest neighbor particles also generate lists containing $W$ and $dW/dr$ for each interacting pair together with lists containing the particle indices.

The final calculation of density can then be carried out using a single loop and the lists of data which is very efficient.

However, this has a drawback of requiring large storage for the lists.

Implementing SPH computer code

The figure shows a typical procedure for SPH simulations.

Initialization
• input of problem geometry
• input of standard parameters

Main SPH process - time integration by predictor corrector or Runge-Kutta.

All sub-functions are called inside the main loop.

Output - save updated data to files, for making videos this could be called at each video output step.
Shock tube in 1-D is a common problem for people starting in SPH simulations. The equations are listed without viscous stress and heat.

Long straight tube filled with gas, which is separated by a membrane in two parts with different pressures and densities but are individually in thermodynamic equilibrium. When the membrane is taken away the following are produced

- a shock wave - moves into region of lower density
- a rarefaction wave (reduction in density) - moves into region of high density
- a contact discontinuity - forms in center and travels into low density region behind the shock
Problem: The shock tube

In this example we use

\[
\begin{aligned}
\frac{D \rho_i}{Dt} &= \sum_{j=1}^{N} m_j (v_i - v_j) \cdot \nabla_i W_{ij} \\
\frac{D v_i}{Dt} &= -\sum_{j=1}^{N} m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} \\
\frac{D e_i}{Dt} &= \frac{1}{2} \sum_{j=1}^{N} m_j (\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij}) (v_i - v_j) \cdot \nabla_i W_{ij} \\
\frac{D x_i}{Dt} &= v_i
\end{aligned}
\]

\[\rho_i = \sum_{j=1}^{N} m_j W_{ij}\]

Or use \(\Pi_{ij} = 0\)

where \(\rho, p, e\) and \(v\) are the density, pressure, internal energy, and velocity respectively.

\(\Delta x\) is the particle spacing.

With 400 particles of the same mass \(m_i=0.001875\).

320 particles evenly distributed in the high density region \([-0.6,0.0]\) and 80 in the low density region \([0.0,0.6]\).
Problem: Equation of State

We use the equation of state for the ideal gas

\[ p = (\gamma - 1) \rho e \]

and the speed of sound is

\[ c = \sqrt{(\gamma - 1)e} \]

\( \gamma = \frac{C_p}{C_v} = 1.4 \) is the ratio of heat capacity

Set the time step to 0.005 and run the simulation for 40 time steps.

No special treatment is used for the boundary as the shock wave has not yet reached it.

Heat capacity the amount of heat required to change a substance's temperature by a given amount
Project assignment

• Implement an SPH code from scratch in matlab to solve the shock wave problem (don't include artificial viscosity yet).

\[
\Pi_{ij} = \begin{cases} 
-\alpha_{\Pi} \bar{c}_{ij} \phi_{ij} + \beta_{\Pi} \phi_{ij}^2 & \text{if } \mathbf{v}_{ij} \cdot \mathbf{x}_{ij} < 0 \\
\bar{\rho}_{ij} & \text{if } \mathbf{v}_{ij} \cdot \mathbf{x}_{ij} \geq 0
\end{cases}
\]

1. Extend the basic implementation to include artificial viscosity of Monaghan ref. [5]. \(\alpha_{\Pi}=1, \beta_{\Pi}=1\) and \(\varphi = 0.1 h_{ij}\) to avoid divergence:

• Reproduce similar results shown for \(\rho, p, e\) and \(v\).

• Write a project report describing, theory, implementation, results.

Help is available from me!

Input data files are also available!
Hints and help for project

Your code should do something like the following:

Input → Time Integration → Output

Single Step

Direct Find
Density \( \rho \)
Internal Force
Art Viscosity
Ext Force, etc.

Kernel

Runga Kutta - calls Single Step

Search for NN pairs

Calculate \( W \) and \( W' \)

Calculate \( \frac{dv}{dt} \) and \( \frac{de}{dt} \)

Calculate the equation of state for \( p \)

Call all subfunctions and sum the results

Calculate the artificial viscosity

Add as many extra modules as needed
The smoothing function (kernel) and its derivative

Use the cubic spline form

\[
W(R, h) = \alpha_d \times \begin{cases} 
\frac{2}{3} - R^2 + \frac{1}{2} R^3 & 0 \leq R < 1 \\
\frac{1}{6} (2 - R)^3 & 1 \leq R < 2 \\
0 & R \geq 2 
\end{cases}
\]

where \( \alpha_d \) is \( 1/h, 15/(7\pi h^2) \), and \( 3/(2\pi h^3) \) and \( \kappa=2 \) in 1-, 2-, and 3-D respectively.

If we take the derivative of this we have

\[
W'(R, h) = \begin{cases} 
\alpha_d \times \left(-2 + \frac{3}{2} R\right) \frac{dx}{h^2} & 0 \leq R < 1 \\
\alpha_d \times \frac{1}{2} (2 - R)^2 \frac{dx}{hr} & 1 \leq R < 2 \\
0 & R \geq 2 
\end{cases}
\]

and

\[
R_{ij} = \frac{r_{ij}}{h} = \frac{|x_i - x_j|}{h} = \frac{|dx|}{h}
\]
Nearest neighboring search algorithms

Use a search algorithm like the following:

```fortran
NIAC=0
DO I=1, NTOTAL
   COUNTIAC(I)=0
END DO
DO I=1,NTOTAL-1
   DO J=I+1, NTOTAL
      IF ( |x(i) - x(j)| ≤ h_i + h_j ) then
         NIAC=NIAC+1
         PAIR_I(NIAC)=I
         PAIR_J(NIAC)=J
         COUNTIAC(I)= COUNTIAC(I)+1
         COUNTJ(J)= COUNTJ(J)+1
      
      Calculate the smoothing function: W(NIAC) for the NIAC-th pair
      Calculate the derivatives: DWDX(D,NIAC) for the NIAC-th pair
   
   END IF
END DO
END DO
```

After storing the PAIR arrays W and DWDX the summations can be done in a single loop as shown below:

```fortran
DO I =1, NTOTAL
   RHO(I)=MASS(I)*W(0)
END DO
DO K=1,NIAC
   I=PAIR_I(K)
   J=PAIR_J(K)
   RHO(I) = RHO(I) + MASS(J)*W(K)
   RHO(J) = RHO(J) + MASS(I)*W(K)
END DO
```
Project 3: Colliding Jupiter like Planets

You can implement this project yourself using the input file: inputPlanet.dat

- First, generalize your code to 3D, $\mathbf{x}$ and $\mathbf{v}$ become vectors and kernel needs additions.
- Implement a gravitational potential in a self consistent manner.

(P. Cossins, Chapter 3, Smoothed Particle Hydrodynamics, Ph.D. Thesis, Leicester 2010, [8])
Gravitational Potential

Use the following equations from the thesis: (P. Cossins, Chapter 3, Smoothed Particle Hydrodynamics, Ph.D. Thesis, Leicester 2010) (Note notation, they use x we use R).

For constant smoothing length the following simplified formula is valid:

\[
\frac{\partial \phi(r, h)}{\partial r} = \begin{cases} 
\frac{1}{h^2} \left( \frac{4}{3} R - \frac{6}{5} R^3 + \frac{1}{2} R^4 \right) & 0 \leq R \leq 1, \\
\frac{1}{h^2} \left( \frac{8}{3} R - 3R^2 + \frac{6}{5} R^3 - \frac{1}{6} R^4 - \frac{1}{15R^2} \right) & 1 \leq R \leq 2, \\
\frac{1}{r^2} & R \geq 2.
\end{cases}
\]

\[
\left( \frac{dv}{dt} \right)_\text{Gravity} ^i = -\frac{G}{2} \sum_j m_j \left( \nabla_i \phi_{ij}(h_i) + \nabla_i \phi_{ij}(h_j) \right)
\]

\[
R_{ij} = \frac{r_{ij}}{h} = \frac{|x_i - x_j|}{h} = \frac{|dx|}{h}
\]
Advanced SPH code
There are some free advanced versions of SPH available on the internet which have been developed for astronomy purposes.
These codes are state of the art and require some effort to install and use.
The following link will take you to a complete list of tools but not all have SPH included:
A widely used SPH tool is Gadget-2 which is a massively parallel cosmological code: http://www.mpa-garching.mpg.de/gadget/
I can probably give some advice about its installation

Conclusions
Hopefully, this has given you a well rounded introduction into the main components of N-body and SPH techniques.