Problem Set 6

Exercises for course Fundamentals of Simulation Methods, WS 2021

Prof. Dr. Mario Flock, Prof. Dr. Friedrich Röpke
Tutors: Brooke Polak (brooke.polak@uni-heidelberg.de | ITA), Glen Hunter (glen.hunter@uni-heidelberg.de | ITA), Jan Henneco (jan.henneco@h-its.org | HITS)
Offices: HITS: Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg; ITA: Albert-Ueberle-Strasse 2, 69120 Heidelberg

Hand in until Wednesday, 24.11., 23:59
Tutorials times: 25.11.
Group 1: Brooke | Thursday 11:00 - 13:00
Group 2: Glen | Thursday 14:00 - 16:00
Group 3: Jan | Friday 11:00 - 13:00

1. Particle-mesh mapping: Weight coefficients and shape functions in 2-D

In the lecture you have learned how to map a set of point particles onto a grid using shape functions. Three shape functions were discussed: a delta function, a top-hat function and a pyramid function. These represent the zeroth, first and second order algorithms. In this exercise you will derive the expressions for the weight coefficients $W_{k,l}(X_i, Y_i)$ for the 2-D case, where k, l are the indices of the cells in x- and y-direction, respectively, and X_i, Y_i are the x- and y-coordinates of particle i. The grid has $K \times K$ cells, equally spaced between $[-H, H]$ in both dimensions. The cell size is therefore $2H/K \times 2H/K$. The symbols (x_k, y_l) represent the cell-centers while $(x_{k\pm1/2}, y_{l\pm1/2})$ represent the cell interfaces. The k, l indices range from 0 to $K - 1$.

1. Given the (X_i, Y_i)-coordinates of particle i, derive an expression for the indices k, l such that $x_{k-1/2} \leq X_i < x_{k+1/2}$ and $y_{l-1/2} \leq Y_i < y_{l+1/2}$, i.e. such that the particle lies inside the cell (k, l).

2. Given how the k and l indices derived in this way for N particles, how would you compute $\rho(x, y)$ function numerically using the zeroth order method? In other words: how would you compute numerically $\rho_{k,l}$ with the Dirac-delta function as the shape function? Either give a function using the definition provided in the lecture notes of the top-hat function or write pseudo-code for this calculation. Note in 2D:

$$\begin{aligned} W_{k,l} &= \int \Pi\left(\frac{\mathbf{X} - \mathbf{x}_p}{h}\right) S(\mathbf{X} - \mathbf{X}_i) d\mathbf{x} \\ &= \int\int \Pi\left(\frac{X - x_k}{h}\right) S(X - X_i) * \Pi\left(\frac{Y - y_l}{h}\right) S(Y - Y_i) dX dY \end{aligned} \tag{1}$$

For the first and second order method, things become a bit more complicated. We will have to define a 3×3 *stencil* of weights

$$W_{k+\delta k, l+\delta l}(X_i, Y_i) \tag{2}$$

with $\delta k = -1, 0, 1$ and $\delta l = -1, 0, 1$. This is a 3×3 matrix with the central element ($\delta k = 0$, $\delta l = 0$) representing the cell containing the particle i, and the surrounding 8 elements are the neighboring cells. We have the normalization

$$\sum_{\delta k=-1,0,+1} \sum_{\delta l=-1,0,+1} W_{k+\delta k, l+\delta l}(X_i, Y_i) = 1 \tag{3}$$

For the zeroth order method this stencil is simple: $W = 0$ for all $(\delta k, \delta l)$ except for $(\delta k, \delta l) = (0, 0)$, for which $W = 1$. In other words: for the zeroth order algorithm the 3×3 matrix has its central element equal to 1, the rest is 0. But for the first and second order methods the 9 elements of the stencil become more complex. Let us write these 9 elements as $W[0,0]$, $W[0,1]$, $W[0,2]$, $W[1,0]$, $W[1,1]$, $W[1,2]$, $W[2,0]$, $W[2,1]$, $W[2,2]$, where we started from 0 instead of -1 simply because most computer languages start array indexing from 0. Note that in Python and C the second of these indices is the x-direction while the first is the y-direction! Now let us define ϵ_x as

$$\epsilon_x = (X_i - x_{k-1/2}) \,/\, (x_{k+1/2} - x_{k-1/2}) \tag{4}$$

(which has the property that $0 \leq \epsilon_x < 1$), and likewise ϵ_y for the y-direction.

3. Derive expressions for the 9 elements of W for the first- and second-order methods. You are allowed to write these expressions in a sequential way: first starting with W[:,:]=1, then doing the x-direction as W[:,{0,1,2}]=W[:,{0,1,2}]**something*, and finally doing the y-direction as W[{0,1,2},:]=W[{0,1,2},:]**something*. To help you on track, we give here the answer for the *second-order* method (in Python), and you have to derive that expression:

```
W[:,:] = np.ones((3,3))
W[:,0]*=0.5-ex+0.5*ex**2
W[:,1]*=0.5+ex-ex**2
W[:,2]*=0.5*ex**2
W[0,:]*=0.5-ey+0.5*ey**2
W[1,:]*=0.5+ey-ey**2
W[2,:]*=0.5*ey**2
```

where `ex` is ϵ_x, and `ey` is ϵ_y. *Note:* Don't forget to also do the first order version. You may need to use an if-statement for that (though it can also be done without).

2. Particle-mesh mapping: Computing a density map from a set of particles in 2-D

Using what we have derived in the previous exercise, we can now put it in practice. We take $H = 10.0$, $K = 20$. We will have N particles of mass M/N, where $M = 1.0$ is the mass of all the particles together.

1. Design and program a function that takes (X_i, Y_i) and returns the indices k, l and the W-matrix. To keep things easy, assume that you can be sure that (X_i, Y_i) are always at least 1 cell width away from the boundary, so that the 3×3 stencil always fits inside the grid.

2. Set up *a single* particle ($N = 1$), randomly positioned in the box, but make sure that it is at least 1 cell away from the boundary. Apply the weighting matrix to compute the density "function" $\rho_{k,l}$ (a $K \times K$ matrix). Show this as an image. Repeat this for zeroth, first and second order methods, and convince yourself that your stencil `W[:,:]` properly places the particle onto the $\rho_{k,l}$ grid.

3. Now set up $N = 100$ randomly positioned points following a 2-D Gaussian probability function with standard deviation $\sigma = 3.0$. Reject all particles that lie within 1 cell of the boundary or beyond the boundary (the number N will thus decrease a bit). Now compute the $\rho_{k,l}$ using the zeroth, first and second order methods, *for the same particle cloud*, and compare these maps.

4. Repeat for $N = 10000$, and convince yourself that this approaches the right answer. For instance: is the integral $\int\int \rho(x, y)\, dx\, dy \simeq M$, as it should (the slight difference being the few particles that were rejected for lying outside the grid)?