ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prof. P. Koumoutsakos, G. Tauriello
ETH Zentrum, CLT F 12, C 11
CH-8092 Zürich

Solution 7

Issued: 13.5.2014

Question 1: Interpolation conserving three moments

We wish to solve the following system for w_1, w_2 and w_3:

$$\begin{aligned} w_1 + w_2 + w_3 &= w_p, \\ w_1\,x_1 + w_2\,x_2 + w_3\,x_3 &= w_p\,x_p, \\ w_1\,x_1^2 + w_2\,x_2^2 + w_3\,x_3^2 &= w_p\,x_p^2. \end{aligned} \tag{1}$$

Taking into account the constant grid spacing we get:

$$w_1 + w_2 + w_3 = w_p, \tag{2}$$

$$w_1\,(x_2 - h) + w_2\,x_2 + w_3\,(x_2 + h) = w_p\,x_p, \tag{3}$$

$$w_1\,(x_2 - h)^2 + w_2\,x_2^2 + w_3\,(x_2 + h)^2 = w_p\,x_p^2. \tag{4}$$

We first use Eq. 2 to simplify Eq. 3 as follows:

$$\begin{aligned} w_p\,x_p &= (w_1 + w_2 + w_3)\,x_2 + w_3\,h - w_1\,h \\ &= w_p\,x_2 + w_3\,h - w_1\,h, \\ w_p\,(x_p - x_2) &= (w_3 - w_1)\,h. \end{aligned} \tag{5}$$

Then we use Eq. 5 and Eq. 2 to simplify Eq. 4:

$$\begin{aligned} w_p\,x_p^2 &= w_1\,(x_2 - h)^2 + w_2\,x_2^2 + w_3\,(x_2 + h)^2 \\ &= w_1\,(x_2^2 + h^2 - 2\,x_2\,h) + w_2\,x_2^2 + w_3\,(x_2^2 + h^2 + 2\,x_2\,h) \\ &= (w_1 + w_2 + w_3)\,x_2^2 + (w_1 + w_3)\,h^2 + (w_3 - w_1)\,2\,x_2\,h \\ &= (w_1 + w_2 + w_3)\,x_2^2 + 2\,w_1\,h^2 + (w_3 - w_1)\,h^2 + (w_3 - w_1)\,2\,x_2\,h \\ &= w_p\,x_2^2 + 2\,w_1\,h^2 + w_p\,(x_p - x_2)(h + 2\,x_2), \\ 2\,w_1\,h^2 &= w_p\,(x_p^2 - x_2^2 - (x_p - x_2)(h + 2\,x_2) \\ &= w_p\,((x_p - x_2)(x_p + x_2) - (x_p - x_2)(h + 2\,x_2) \\ &= w_p\,(x_p - x_2)(x_p + x_2 - h - 2\,x_2) \\ &= w_p\,(x_p - x_2)(x_p - x_2 - h), \\ w_1 &= \frac{1}{2}\,w_p\,\lambda\,(\lambda - 1), \end{aligned} \tag{6}$$

where we used $\lambda = (x_p - x_2)/h$. We can now insert Eq. 6 into Eq. 5 to get:

$$\begin{aligned} w_p\,(x_p - x_2) &= (w_3 - w_1)\,h, \\ w_p\,\lambda &= w_3 - w_1 \\ &= w_3 - \frac{1}{2}\,w_p\,\lambda\,(\lambda - 1), \\ w_3 &= w_p\,\lambda\,(1 + \frac{1}{2}\,\lambda - \frac{1}{2}) \\ &= \frac{1}{2}\,w_p\,\lambda\,(\lambda + 1). \end{aligned} \tag{7}$$

Finally we insert Eq. 6-7 into Eq. 2 to obtain:

$$\begin{aligned} w_1 + w_2 + w_3 &= w_p, \\ w_2 &= w_p - \frac{1}{2}\,w_p\,\lambda\,(\lambda - 1) - \frac{1}{2}\,w_p\,\lambda\,(\lambda + 1) \\ &= w_p\,(1 - \lambda^2). \end{aligned} \tag{8}$$

In order to get f_0, f_1 and f_2, and therefore the Λ_2 kernel, we introduce $\lambda_i = (x_p - x_i)/h$ (i.e. $\lambda_1 = \lambda + 1$, $\lambda_2 = \lambda$ and $\lambda_3 = \lambda - 1$) and write:

$$w_1 = \frac{1}{2}\,w_p\,(\lambda_1 - 1)\,(\lambda_1 - 2), \tag{9}$$

$$w_2 = w_p\,(1 - \lambda_2^2), \tag{10}$$

$$w_3 = \frac{1}{2}\,w_p\,(\lambda_3 + 1)\,(\lambda_3 + 2), \tag{11}$$

or write w_i as a function of λ_i assuming that x_2 is the closest grid point to x_p (i.e. $\lambda \in [-0.5, 0.5]$) as:

$$w_i = w_p \begin{cases} \frac{1}{2}\,(\lambda_i - 1)\,(\lambda_i - 2), & \text{if } 0.5 < \lambda_i \leq 1.5, \\ (1 - \lambda_i^2), & \text{if } -0.5 \leq \lambda_i \leq 0.5, \\ \frac{1}{2}\,(\lambda_i + 1)\,(\lambda_i + 2), & \text{if } -1.5 \leq \lambda_i < -0.5, \\ 0, & \text{else.} \end{cases} \tag{12}$$

Question 2: Implement particle to mesh interpolation scheme

Running the complete code with the parameters given, we observe that indeed the 0th, 1st and 2nd moments are conserved while the 3rd is not. The relevant code snippet are listed and commented hereafter.

Λ_2 **constructor**:

```
1 Lambda2()
2 : support_start(-1), support_end(2), support(support_end-support_start
      )
3 {
4 }
```

The support of the Λ_2 kernel goes [-1.5,1.5[, thus only three grid points are needed: the one closest to the particle, one to its left and one to its right. The support in index space with respect to the closest mesh point is then $\{-1, 0, 1\}$, and so `support_start` and `support_end`, the loop bounds, are -1 and $+2$ respectively.

Λ_2 **kernel**:

```
double kernel(double x)
{
    // TODO: write kernel for the interpolation
    double absx = fabs(x);

    if (absx<1.5 && absx>=.5)
        return .5*(2.-absx)*(1.-absx);
    else if (absx<.5)
        return 1. - absx*absx;
    else // outside of support
        return 0.;
}
```

Due to the symmetry of the kernel we are able to rewrite the Λ_2 kernel with respect to $|\lambda_i|$:

$$w_i = w_p \begin{cases} \frac{1}{2}\left(|\lambda_i| - 1\right)\left(|\lambda_i| - 2\right), & \text{if } 0.5 \leq |\lambda_i| < 1.5, \\ (1 - \lambda_i^2), & \text{if } |\lambda_i| < 0.5 \end{cases} \tag{13}$$

Note that the kernel is expressed in function of λ and not the physical distance given by $x_p - x_i$.

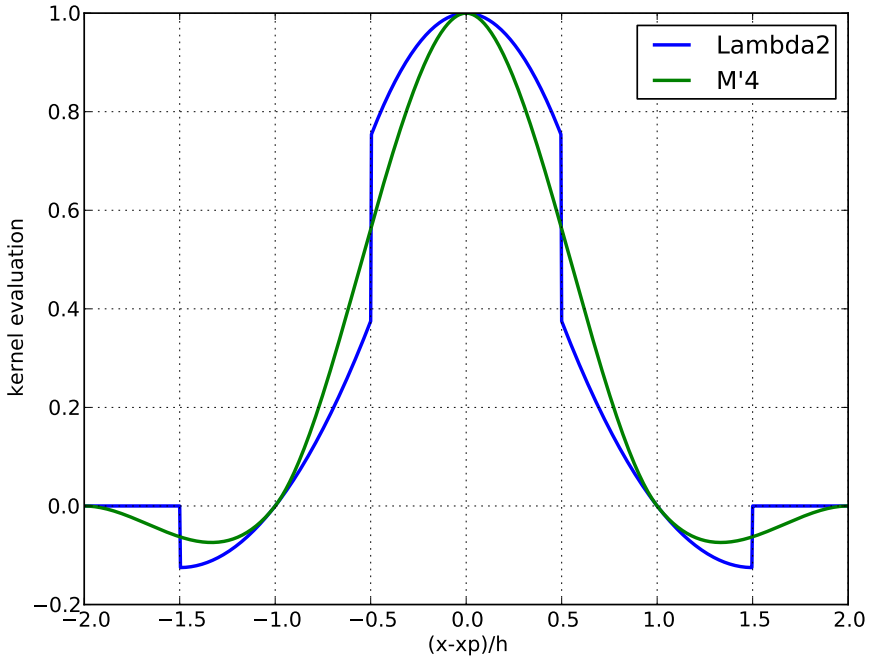
Λ_2 **p2m**:

```
// perform 2 dimensional P2M of mass contained in particles to the
    mesh
void p2m(vector<Particle>& particles, Grid2D& mesh)
{
    // TODO:
    //  1. make sure all mesh values are set to 0
    //  2. for each particle find the closest mesh point indices
    //  3. compute the weights for the interpolation
    //  4. perform the P2M operation using the weights
    const double h = mesh.getH();

    // since we are accumulating, we have to make sure the mesh is
        empty
    mesh.clear();

    for (int p=0; p<particles.size(); p++)
    {
        // find the two closest mesh point indices
        const double px = particles[p].x/h;
        const double py = particles[p].y/h;
        int idx = (px-floor(px))<.5 ? (int)floor(px) : (int)ceil(px);
        int idy = (py-floor(py))<.5 ? (int)floor(py) : (int)ceil(py);

        // compute weights
        double wx[support], wy[support];
        for (int i=support_start; i<support_end; i++)
```

```
25          {
26              wx[i-support_start] = kernel(px-(idx+i));
27              wy[i-support_start] = kernel(py-(idy+i));
28          }
29
30          // distribute mass
31          for (int j=support_start; j<support_end; j++)
32              for (int i=support_start; i<support_end; i++)
33                  mesh(idx+i,idy+j) += wx[i-support_start]*wy[j-
                        support_start]*particles[p].m;
34      }
35  }
```

As the Λ_2 kernel uses an odd number of mesh points, we need to find the closest one and take its two neighbors. The closest point is the one for which $|\lambda_i| < 0.5$, as the next closest one would be at $\lambda_j = 1 - \lambda_i$.

We see that both kernels conserve the 0th, 1st and 2nd moments, whereas the 3rd moment (the last four numbers printed) is not necessarily conserved. The accuracy of the kernels is therefore comparable, and the Λ_2 kernel uses only 3 gridpoints for each particle whereas the M_4' kernel uses 4. This means the Λ_2 kernel has a significant lower computational cost for the same accuracy, yet in practice the M_4' kernel is used more frequently. The reason for this is visible when we plot the two kernels:



The Λ_2 kernel, shown with the blue curve, has discontinuities at $(x - x_p)/h = \pm 0.5$ and $(x - x_p)/h = \pm 1.5$, whereas the M_4' kernel is smooth throughout its support. This smoothness is realized at the cost of an extra point in its support. In a numerical setting, non-smoothness is often avoided because it can cause jumps in a relevant field, which in turn can cause oscillations when, for instance, a high order finite difference scheme is used on the interpolated field. This is an important reason for the popularity of the M_4' kernel in numerical simulations, even though its usages comes at a higher cost than Λ_2.