

CMIS Hand-in 2: Finite Difference Methods 2

Nikolai Plambeck Nielsen

lpk331@alumni.ku.dk

Niels Bohr Institute, University of Copenhagen

1 INTRODUCTION

2 SEMI-LAGRANGIAN IMPLICIT TIME INTEGRATION: SOLVING THE ADVECTION EQUATION

In this problem we encounter the advection equation:

$$\frac{D\phi}{Dt} = (\mathbf{u} \cdot \nabla)\phi + \frac{\partial\phi}{\partial t} \quad (1)$$

Where $D\phi/Dt$ is the particle derivative of the scalar field ϕ , and \mathbf{u} is the associated velocity field. We set $D\phi/Dt = 0$ such that no dissipation occurs for the system:

$$\frac{\partial\phi}{\partial t} = -(\mathbf{u} \cdot \nabla)\phi \quad (2)$$

The advection equation describes the flow of some scalar field ϕ in a velocity field \mathbf{u} . This could for example describe the motion of food colouring in a pool of water. Then ϕ would be the density of the food colouring molecules, whilst \mathbf{u} would be the flow of the water in the pool.

The idea behind semi-lagrangian time integration is to use this property of the advection equation, and directly use the values of the scalar field at the previous times step. Specifically we treat our grid points as particles floating along the vector field. At a previous time, the grid point particles (to first order) were at

$$\mathbf{x}^{t-\Delta t} = \mathbf{x}^t - \Delta t \mathbf{u} \quad (3)$$

we then use the the value of the field at these points as the value for ϕ on the grid points, for the next time step:

$$\phi(\mathbf{x}^t) \leftarrow \phi(\mathbf{x}^{t-\Delta t}) \quad (4)$$

The problem then, is that we are not guaranteed, that the grid point particles hit a grid point position, and as such we might not know the value of $\phi(\mathbf{x}^{t-\Delta t})$. To get around this we use a bilinear interpolation between the four nearest grid points to $\mathbf{x}^{t-\Delta t}$, and use this interpolated value as the true value.

For this problem we use a velocity field of $\mathbf{u} = (y, -x)^T$, corresponding to constant circular motion in the clockwise direction. For the scalar field we use a sum of Gaussian functions. If we choose the width of the Gaussians to be sufficiently small, and a large domain, virtually all the points of ϕ that are non-zero will be nowhere near the boundary of the domain. This ensures that a bilinear interpolation always has enough points to use for the interpolation.

With this vector field, we expect the scalar field to rotate in time, with a period of 2π . We can then perform an (approximate) full rotation in N steps by defining $\Delta t = 2\pi/N$. It is only approximate due to the truncation error present when discretising the system.

2.1 Experiments

For this problem I perform two experiments. In the first I want to test how the rotation depends on the time step, and in the second

N	$\text{atan2}(x'_1, x'_2)$	$ \mathbf{x}' / \mathbf{x} $
74	-0.997	1.300
207	-0.0323	1.090

Table 1: Number of iterations per revolution needed to achieve an accuracy greater than 0.1 in both argument and ratios of magnitudes.

I test how much bleeding occurs, again as a function of the time step and grid spacing.

To test the rotation, I take a point, say $\mathbf{x} = (2, 0)^T$ and backtrace N times to get \mathbf{x}' , corresponding to one rotation of the whole grid, and compare the magnitude and argument of \mathbf{x}' to that of \mathbf{x} . An example of the trajectory is seen in figure **INSERT REF**, with $\Delta t = \pi/25 \approx 0.126$. In figure **INSERT REF** the ratio of magnitudes $||\mathbf{x}'||/||\mathbf{x}||$ and the argument of \mathbf{x}' ($\text{atan2}(x'_1, x'_2)$) is calculated for a range of simulations. As seen in the figure, the values tend to 1 and 0 respectively, corresponding to:

$$\lim_{\Delta t \rightarrow 0} \mathbf{x}' = \mathbf{x}. \quad (5)$$

To number of iterations per rotation needed to achieve an accuracy of 0.1 in the argument and ratios of magnitude are tabulated in table 1.

3 MEAN CURVATURE FLOW WITH FINITE DIFFERENCE METHODS

Next we turn our attention to solving the mean curvature flow equation for some signed distance field. Mean curvature flow is analogous to wrapping a tight string around an isosurface of a scalar field and then tightening the string such that it follows the isosurface at a higher value. The equation is given by

$$\frac{\partial\phi}{\partial t} = \nabla \cdot \frac{\nabla\phi}{||\nabla\phi||} = \kappa \quad (6)$$

The right hand side of the equation can be rewritten as

$$\kappa = \frac{\nabla\phi^T \nabla\phi \text{tr } H - \nabla\phi^T H \nabla\phi}{||\nabla\phi||^3} \quad (7)$$

with H being the Hessian matrix.

3.1 Signed Distance Fields

A signed distance field is a scalar field describing the signed euclidian distance from each pixel of a black and white, binary image, to the nearest border between black and white, in units of pixels. The sign is such that white pixels are counted as inside, with a negative distance to the border. A one dimensional is seen in figure **INSERT REFERENCE**, where the inside is defined to be $x \in [5, 11]$. For points sufficiently close to a border will have a gradient of approximately 1.

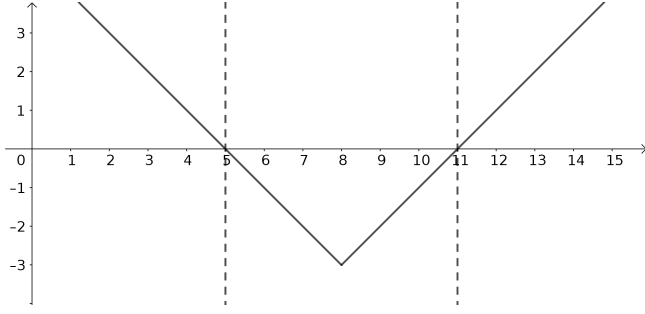


Figure 1: A signed distance field in one dimension, with the inside as $x \in [5, 11]$.

3.2 Discretisation

For the domain we employ a central difference approximation for each of the different spatial derivatives. This gives

$$\kappa \approx k = \frac{(D_x \phi)^2 D_{yy} \phi + (D_y \phi)^2 D_{xx} \phi - 2 D_{xy} \phi D_x \phi D_y \phi}{[(D_x \phi)^2 + (D_y \phi)^2]^{3/2}} \quad (8)$$

where D is the central difference approximation, where the subscripts denote which variables are differentiated with respect to. For example the central difference approximation for the second order derivative with respect to x :

$$D_{xx} \phi_{ij} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2}. \quad (9)$$

Some problems with this equation would be at or around points where the norm of the gradient disappears. This would result in possible division by zero, or just a very large value of k .

A conservative maximum bound for the mean curvature is given by the inverse of the grid spacing: $k_{\max} = \max(\Delta x, \Delta y)^{-1}$. To remedy values that are too large one can clamp the value of k to between $\pm k_{\max}$ by

$$k \leftarrow \max(-k_{\max}, \min(k, k_{\max})). \quad (10)$$

For the temporal derivative we employ a first order forward difference to integrate in time:

$$\phi_{ij}^{t+\Delta t} = \phi_{ij}^t + \Delta t k_{ij}, \quad (11)$$

for some small Δt .

This numerical scheme is conditionally stable, since too large a time step will lead to numerical blow up of the solution. The condition is called the CFL condition, and is given by

$$\Delta t \leq \frac{h}{2k_{\max}}, \quad h = \min(\Delta x, \Delta y) \quad (12)$$

3.3 Boundary conditions

For a signed distance field, the boundary nodes will not have a fixed value, and as such a von Neumann boundary condition is preferable to a Dirichlet boundary condition. In general, the signed distance field will have a gradient of ± 1 in each direction, so long as the node is sufficiently far away from the midpoint of a region. Since midpoints cannot occur at the boundary, it makes sense to extend the derivative to the boundary. For the left boundary this would

mean $d\phi_{1,j}/dx = d\phi_{2,j}/dx$. This is implemented using ghost nodes and a central difference approximation:

$$\frac{\partial \phi_{1,j}}{\partial x} = \frac{\partial \phi_{2,j}}{\partial x} \Rightarrow \frac{\phi_{2,j} - \phi_{0,j}}{2\Delta x} = \frac{\phi_{3,j} - \phi_{1,j}}{2\Delta x} \quad (13)$$

Giving the updating formula for ghost nodes on the left boundary:

$$\phi_{0,j} = \phi_{1,j} + \phi_{2,j} - \phi_{3,j}. \quad (14)$$

3.4 Experiments