1. 1-D PM code

In the language of your choice (IDL, Mathematica, c, ...), write a 1-dimensional gravitational Particle-Mesh (PM) code. Choose whatever units of length, mass, and time make programming easiest.

(a) Functions

Your code should contain functions that implement the following:

1. A cloud-in-cell (CIC) smoothing window \( W(x) \);
2. Wavenumber of a Fourier mode;
3. Mesh density given particle positions;
4. FFT of mesh density;
5. Potential given density, in Fourier space;
6. Acceleration given potential, in Fourier space;
7. FFT acceleration back on to mesh;
8. Acceleration at particle positions given mesh acceleration.

[Hints: Function 2 depends on how the FFT you use stores the FT. In mathematica for example, the FT is stored as a complex array \( \{\tilde{a}_0, \tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_{[N/2]}, \tilde{a}_{[N/2]+1-N}, \ldots, \tilde{a}_{-2}, \tilde{a}_{-1}\} \), so that the wavenumbers are \( \{0, 1, 2, \ldots, [\frac{N}{2}], [\frac{N}{2}] + 1 - N, \ldots, -2, -1\} \). FFTs that input a real array typically return \( \{\tilde{a}_0, \text{Re } \tilde{a}_1, \text{Im } \tilde{a}_1, \ldots, \text{Re } \tilde{a}_{[N/2]}\} \), in which case the wavenumbers are \( \{0, 1, 1, \ldots, [\frac{N}{2}]\} \).

Function 5 should set the zero'th Fourier mode of the potential to zero, \( \tilde{\phi}_0 = 0 \). Function 6 should set the zero'th and \( [\frac{N}{2}]' \)th (for even \( N \)) Fourier mode of the acceleration to zero, \( \tilde{g}_0 = \tilde{g}_{[N/2]} = 0 \). The sign of the acceleration \( \tilde{g} \) in Fourier space depends on the phase convention of the FFT; for example mathematica’s phase convention is minus the convention adopted in class.]

(b) Leap-frog integrator

Implement a leap-frog integrator for the positions \( x \) and velocities \( v \) of particles:

\[
\begin{align*}
v_{i+\frac{1}{2}} &= v_{i-\frac{1}{2}} + g_i \Delta t, \\
x_{i+1} &= x_i + v_{i+\frac{1}{2}} \Delta t,
\end{align*}
\]

where \( \Delta t \) is a suitable time step.

(c) Integrate and plot

Choose a reasonable mesh-size \( N \). Choose a small number of particles, say 2 to 4. Give the particles random initial positions, and zero initial velocities. Choose a time step \( \Delta t \). Integrate for a sufficient number of steps, at least 100. Plot the positions of the particles as a function of time \( t \).