Velocity Autocorrelative Function Calculation

As the lecture 1 shown, the Brownian motion is one kind of Markov processes, so those partials move randomly at each times. Although its location function A(t) is changing randomly, for two measurements taken at times t and t_{corr}, which both come from the same data set but the second one is shifted by time t_{corr}, there are good chances that A(t') and A(t_{corr}) have the similar values, so we call their values are autocorrelated.

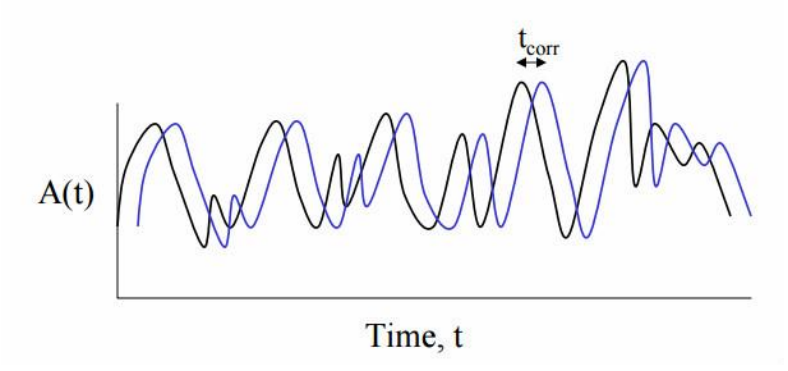

Fig1. The location function A(t) and its deformation A(t_{corr}).[1]

As for the autocorrelative function, a single number α is given to represent the average of the multiply value between these two data set over the whole time range.

$$\text{VAF} = \langle A(t)A(tcorr)\rangle \qquad \text{eq.1}$$

For a huge N numbers of partials, it is calculated as the followed equation shown, which is also substituted by the velocity values.

$$\text{VAF} = \frac{1}{N}\sum_{i=1}^{N}[vi(t)vi(t+\Delta t)+vj(t)vj(t+\Delta t)] \qquad \text{eq.2}$$

The calculation is achieved by following these steps and the relevant code:

I. Set a random array
II. Compute the mean
III. Compute the variance
IV. Compute the autocorrelation
V. Code

```
int main(){
            double x, y;
            double u, v;
            int step;
            FILE *fout;
            int i, j;
            int count[total_step];
            double factor;
            double vaf[total_step];
            static double velx[total_step], vely[total_step];
            x = 0.0; y = 0.0;
            u = 0.0; v = 0.0;
            for (step = 0; step < total_step; step++){
                  x += delta_t * u;
                  y += delta_t * v;
                  u += delta_t / mass * (-gamma * u + fr());
                  v += delta_t / mass * (-gamma * v + fr());
                  velx[step] = u;
                  vely[step] = v;}
            for (step = 0; step < out_step; step++) {
                  count[step] = 0;
                  vaf[step] = 0;}
            for (i = 0; i < total_step; i++) {
                  for (j = i; j < total_step; j++){
                      step = j - i;
                      count[step]++;
                      vaf[step] += (velx[i]*velx[j] + vely[i]*vely[j]);}}
            factor = vaf[0]/count[step];
            fout = fopen("vaf.dat", "w");
            for (step = 0; step < out_step; step++) /{
                  fprintf(fout, "%10.3f %10.3f \n",
                    delta_t * step, vaf[step] / count[step] / factor);}
            fclose(fout);
            return 0;}
```

The code is conducted for times by changing the value of Δt from 0.1 to 0.7. And here are the original figures and their scaling version based on the calculation results.

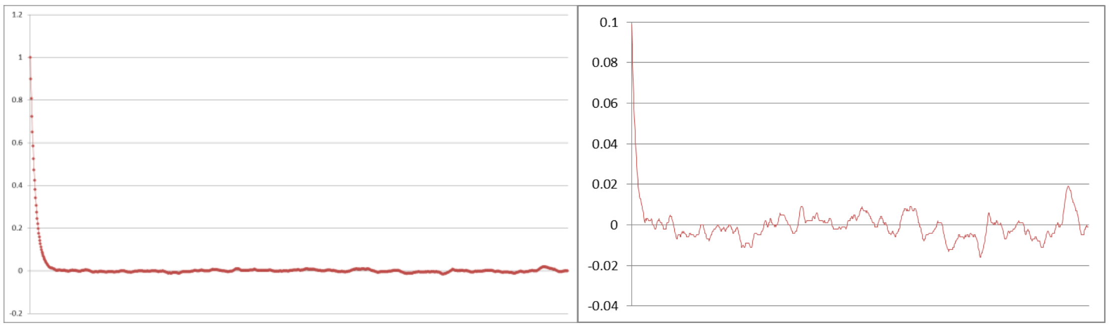

Fig.2 Δt = 0.1

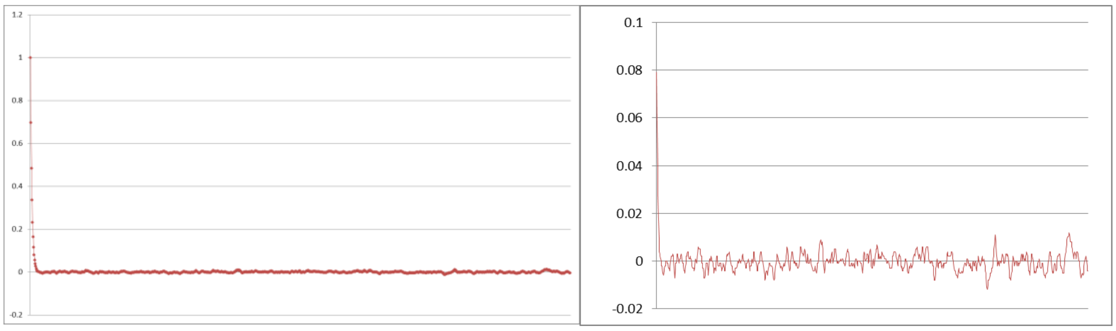

Fig.3 Δt = 0.3

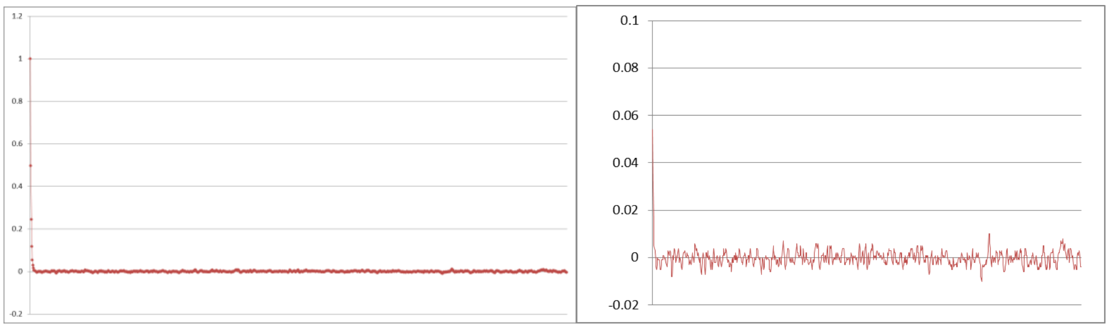

Fig.3 Δt = 0.5

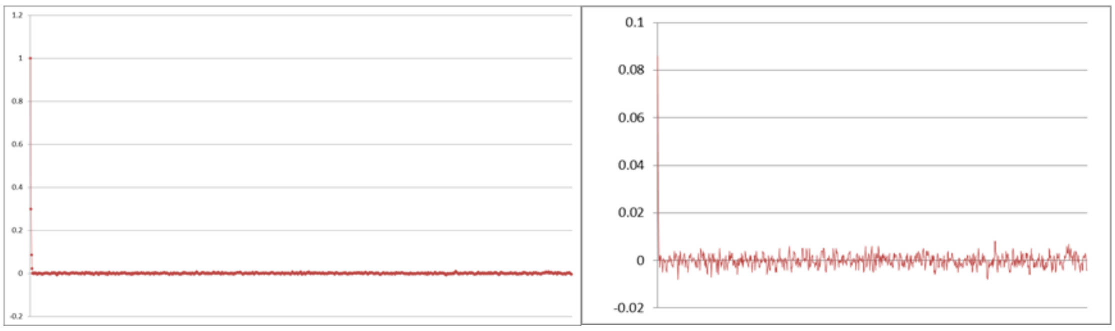

Fig.4 Δt = 0.7

Bibliography:

[1] Introduction to Atomistic Simulations. Leonid Zhigilei. University of Virginia

[2] Program to compute autocorrelation for a series X of size N. Kenneth J. Christensen. University of South Florida