MD-Assignment 3

Carlos Pereyra

May 2019

1 Assignment

The following document will outline the theory of the HMO with noise and damping (Langevin relation) and then the proceeding sections will demonstrate resulting plots.

1.1 Tasks

1. Generate 10^{10} random Gaussian numbers from two uniformly distributed variables.
2. Show average total energy as a function of time step (dt).
   - "Calculate averages (over integer number of periods) for Potential, Kinetic (U,V), and total energies as a function of normalized time-step"
   - So basically sum both kinetic and potential energies each time step and divide by the total number of time steps at the end - then switch the friction coefficient.
   - visualize displacement r(t) for fun
3. For the particle in a box simulation - "Open the boundaries — i.e., remove the walls from your interactions."
   - Instead, implement periodic boundaries as follows: For each interaction you calculate the vector distance (dx,dy) between any two particles. Check the following:
   - if dx >= 0.5*Lx then dx -= Lx
   - if dx < -0.5*Lx then dx += Lx
   - Do the same for dy. Then compute the forces and proceed.
   - after each time step you should check the following for each particle:
   - if x >= Lx then x -= Lx
   - if x < 0 then x += Lx
   - Do the same for y. This procedure now simulates a bulk system, where all particles interact with each other's closest image in a periodic lattice with periodicity, Lx, Ly.

2 Task 1: Langevin and Noise Origins

Algorithms and relations needed for this assignment.

2.1 Box-Muller Algorithm

The Box-Muller Algorithm (1 and 2) relies on two numbers (ξ_1 and ξ_2) that are taken from a uniform distribution.

$$\eta_1 = \sqrt{-2\ln(\xi_1)}\cos(2\pi\xi_2) \tag{1}$$

$$\eta_2 = \sqrt{-2\ln(\xi_1)}\sin(2\pi\xi_2) \tag{2}$$

We can then use these two numbers (ξ_1 and ξ_2) to compute a Gaussian distributed value (η_1 or η_2). If we plot the distribution, by creating a histogram, of (1) and (2) we should see a Gaussian distribution. Regardless of the quality of random number generator, the distribution should look Gaussian, a study of the HMO's response will provide a better view of the pseudo-random number generator's quality.

$$g(\eta_1,\eta_2) = -\frac{1}{2\pi}e^{-\frac{\eta_1^2}{2}}e^{-\frac{\eta_2^2}{2}} \tag{3}$$

$$= -\frac{1}{2\pi}e^{-\left(\frac{\eta_1^2+\eta_2^2}{2}\right)} \tag{4}$$

$$= -\frac{1}{2\pi}e^{-\frac{1}{2}\left(\eta_1^2+\eta_2^2\right)} \tag{5}$$

$$= -\frac{1}{2\pi}e^{-\frac{1}{2}\left(-2\ln(\xi_1)\cos^2(2\pi\xi_2)+-2\ln(\xi_1)\sin^2(2\pi\xi_2)\right)} \tag{6}$$

$$= -\frac{1}{2\pi}e^{\ln\xi_1\left[\cos^2(2\pi\xi_2)+\sin^2(2\pi\xi_2)\right]} \tag{7}$$

$$= -\frac{1}{2\pi}e^{\ln\xi_1} \tag{8}$$

$$= -\frac{1}{2\pi}\xi_1 \tag{9}$$

What does (9) actually mean? I do not know. But in the meantime, lets take a look at the produced data for the ξ_1, ξ_2, ν_1, and ν_2 distributions (Figure 1 and 2). Figures 1 and 2) demonstrate n=10^4; meanwhile, n=10^8 are demonstrated in Figures (3 and 4). Generating 10^8 uniformally distributed samples (Figures 3 and 4) took approximately ten hours, even with the sampling within four sigma's or four standard deviations method.

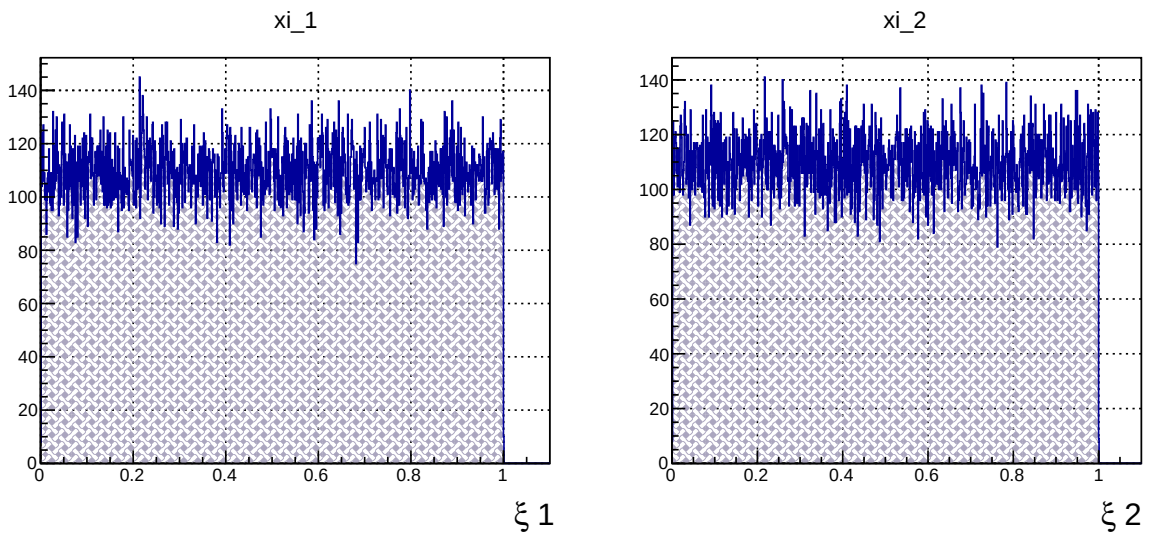

Figure 1: Uniform ξ_1 and ξ_2 Distribution with n=10^4

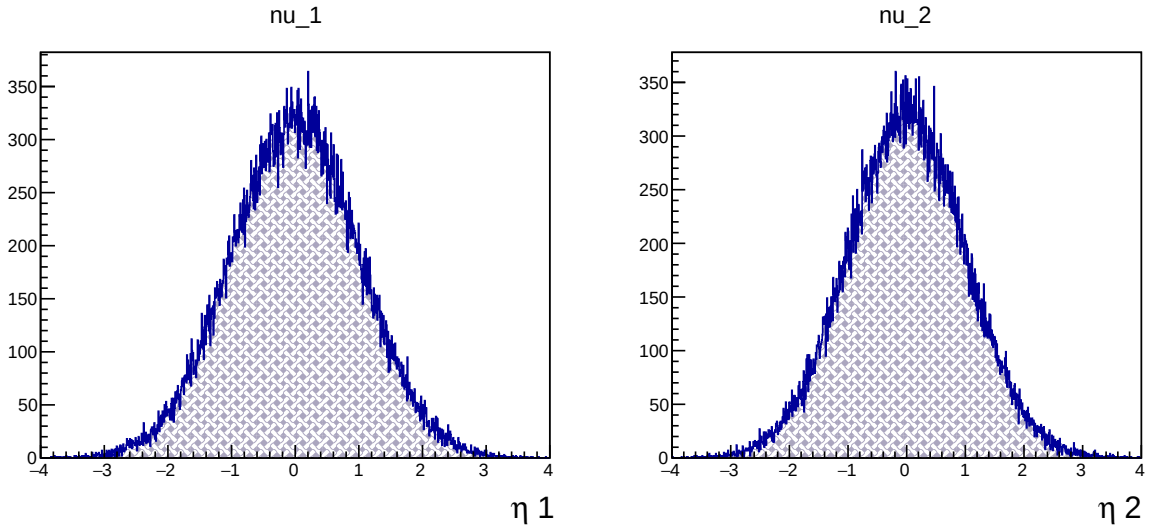

Figure 2: Gauss η_1 and η_2 Distribution with n=10^4

We then use our choice of η_1 or η_2 to determine $\beta^{n+1} = \sqrt{2\alpha kTdt}\eta$.

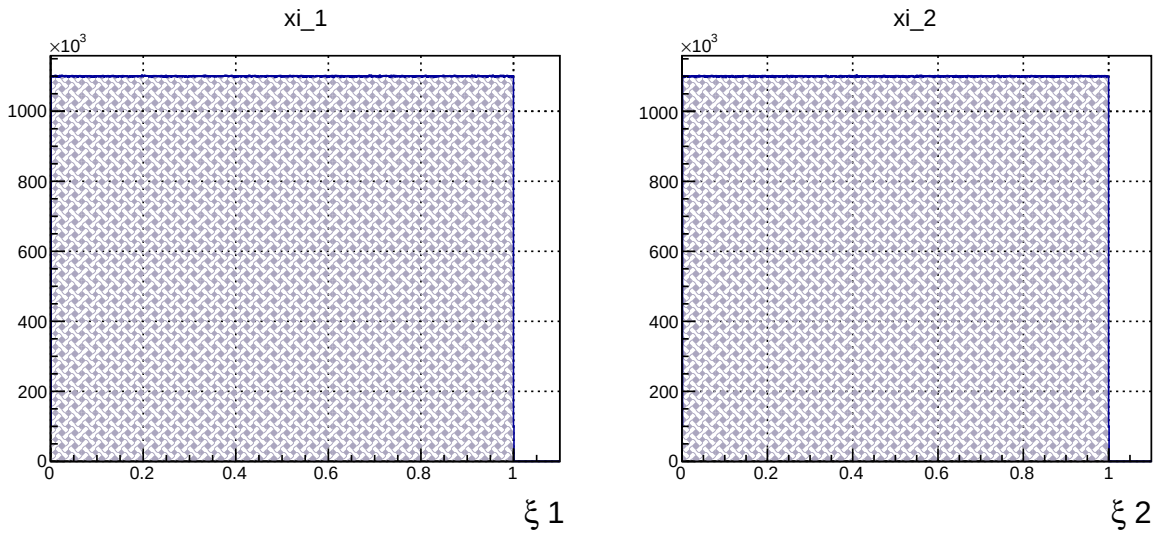

Figure 3: Uniform ξ_1 and ξ_2 Distribution with n=10^8

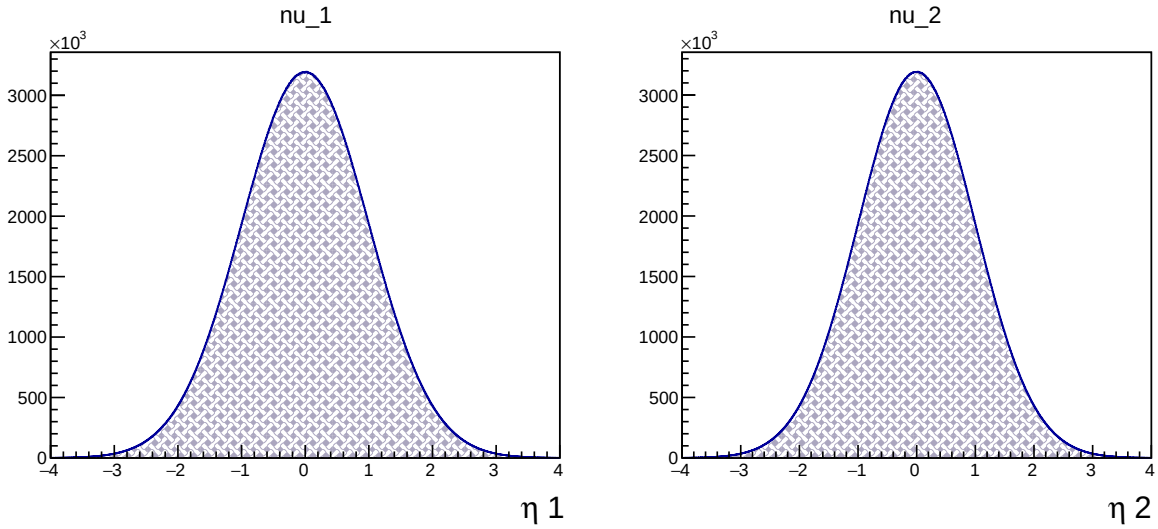

Figure 4: Gauss η_1 and η_2 Distribution with n=10^8

3 Task 2: Modeling Langevin for a HMO System

The most common way of bench marking the stability of a discritized ODE solution is to model the energy as a function of time step within a harmonic oscillatory (HMO) system. We will now model the Langevin HMO with friction and stochastic noise. We start with the Langevin equation (Equation 10) and show the friction and noise terms. The following derivation illustrates discrete linearization, which we can then use to compute inside any scripting language.

$$m\ddot{r} + \alpha\dot{r} = f + \beta \tag{10}$$

Let us rewrite the Langevin equation in a more convenient fashion.

$$m\ddot{r} = f + \beta - \alpha\dot{r}$$

Now we begin to discretize by taking the integral.

$$m\int_{t_n}^{t_{n+1}} \dot{v}dt = \int_{t_n}^{t_{n+1}} fdt + \int_{t_n}^{t_{n+1}} \beta(\xi)dt - \alpha\int_{t_n}^{t_{n+1}} \dot{r}dt \tag{11}$$

Our goal is to determine $(r^{n+1} - r^n)$ and $(v^{n-1} - v^n)$. In order to do this we utilize the trapezoidal approximation shown below (12 and 13), let us define some things that will help us find $(r^{n+1} - r^n)$ and $(v^{n-1} - v^n)$.

$$\int_{t_n}^{t_{n+1}} \dot{r}dt \approx \frac{dt}{2}\left(v^{n+1} + v^n\right) \qquad \int_{t_n}^{t_{n+1}} \dot{r}dt \Rightarrow r^{n+1} - r^n \tag{12}$$

$$\int_{t_n}^{t_{n+1}} fdt \approx \frac{dt}{2}\left(f^{n+1} + f^n\right) \tag{13}$$

Let us write the langevin equation (10) in discrete form.

$$m\left[v^{n+1} - v^n\right] = \int_{t_n}^{t_{n+1}} fdt + \beta_{n+1} - \alpha\left[r^{n+1} - r^n\right] \tag{14}$$

$$\left[v^{n+1} - v^n\right] = \frac{1}{m}\left[\int_{t_n}^{t_{n+1}} fdt + \beta_{n+1} - \alpha\left[r^{n+1} - r^n\right]\right] \tag{15}$$

To find $(r^{n+1} - r^n)$ we remember 12 so basically:

$$\begin{aligned} r^{n+1} - r^n &\approx \frac{dt}{2}\left(v^{n+1} + v^n\right) \\ &\approx \frac{dt}{2}\left(v^{n+1} - v^n + 2v^n\right) \end{aligned}$$

We can then replace $v^{n+1} - v^n$ with our equation 15.

$$r^{n+1} - r^n \approx \frac{dt}{2}\left(\frac{1}{m}\left[\int_{t_n}^{t_{n+1}} fdt + \beta_{n+1} - \alpha\left[r^{n+1} - r^n\right]\right] + 2v^n\right) \tag{16}$$

Do more simplifications to the equation above and define $b = \frac{1}{1+\frac{\alpha dt}{2m}}$. We end up with equation 17.

$$\boxed{r^{n+1} = r^n + b\left[dtv^n + \frac{dt^2}{2m}f^n + \frac{dt}{2m}\beta^{n+1}\right]} \tag{17}$$

Similarly to get $v^{n+1} - v^n$ we go back to the Langevin equation (10) and write it discretely.

$$m\left(v^{n+1} - v^n\right) = \int_{t_n}^{t_{n+1}} fdt + \beta^{n+1} - \alpha\left(r^{n+1} - r^n\right)$$

$$m\left(v^{n+1} - v^n\right) = \frac{dt}{2}\left(f^{n+1} - f^n\right) + \beta^{n+1} - \alpha b\left[dtv^n + \frac{dt^2}{2m}f^n + \frac{dt}{2m}\beta^{n+1}\right]$$

yada yada...

Then we do more simplification and arrive at equation 18.

$$\boxed{v^{n+1} = av^n + \frac{dt}{2m}\left(f^{n+1} + af^n\right) + \frac{b}{m}\beta^{n+1}} \tag{18}$$

We can do further simplifications and rewrite (r^{n+1}) and (v^{n-1}) with the half step velocity, which I have not formally derive myself. But here's the algorithm produced by Dr. Niels Jensen.

$$\begin{aligned}
u^{n+1} &= \sqrt{b}\left[v^n + \frac{dt}{2m}f^n + \frac{1}{2m}\beta^{n+1}\right] \\
r^{n+1} &= r^n + \sqrt{b}dtu^{n+1/2} \\
v^{n+1} &= \frac{a}{\sqrt{b}}u^{n+1/2} + \frac{dt}{2m}f^{n+1} + \frac{1}{2m}\beta^{n+1} \\
\beta^{n+1} &= \sqrt{2\alpha kTdt}\sigma^{n+1} \\
&= \sqrt{2\alpha kTdt}\eta_1 \\
a &= \frac{1 - \frac{\alpha dt}{2m}}{1 + \frac{\alpha dt}{2m}} \\
b &= \frac{1}{1 + \frac{\alpha dt}{2m}}
\end{aligned}$$

don't forget that we will be modeling a HMO so:

$$f = -kr^n$$

and

$$\Omega_0 dt = \sqrt{\frac{k}{m}}dt$$

3.1 Modeling the Langevin

Results of the discrete Langevin equation with a HMO potential are illustrated in Figures 5 and 6. The first pad in Figure represents no damping due to alpha being set to zero, which also seems to illustrate no noise from the β^{n+1} term (this probably should not be happening).

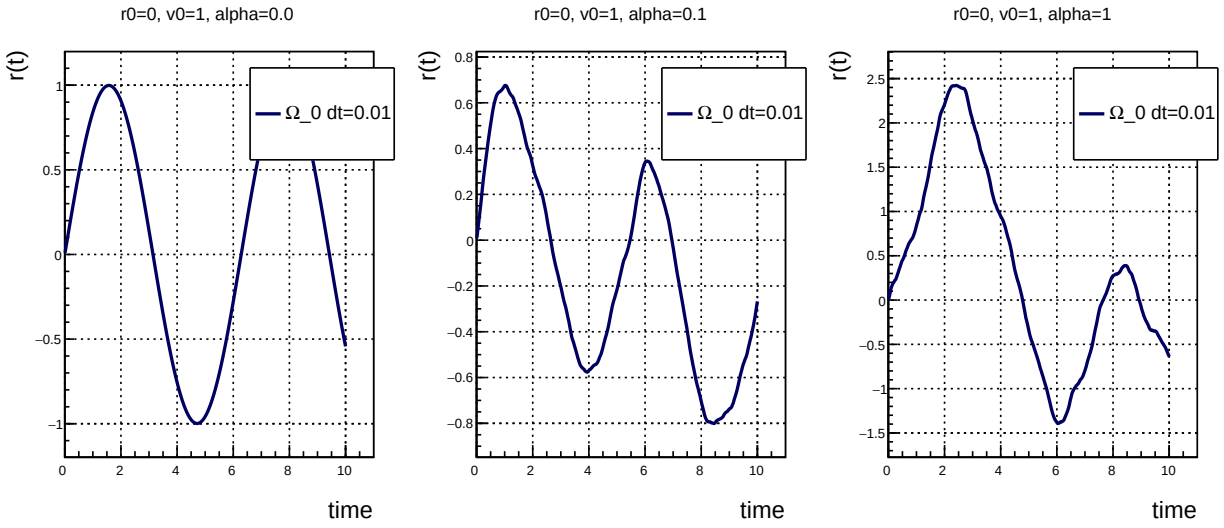

Figure 5: Damping visualization, initial conditions r_0=1 v_0=0

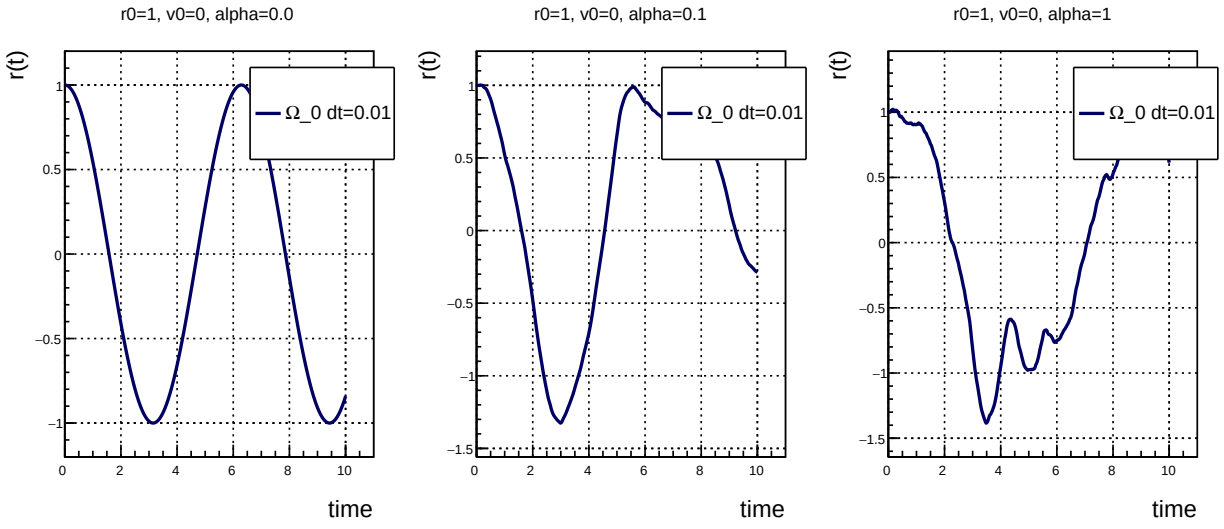

Figure 6: Damping visualization, initial conditions r_0=1 v_0=0

3.2 Temperature=0.1

Figures 7 and 8 represent averaged energies as a function of time-step interval (dt) with a total of 10^3 time steps. **Test conditions** are included in the title heading of each plot pad.

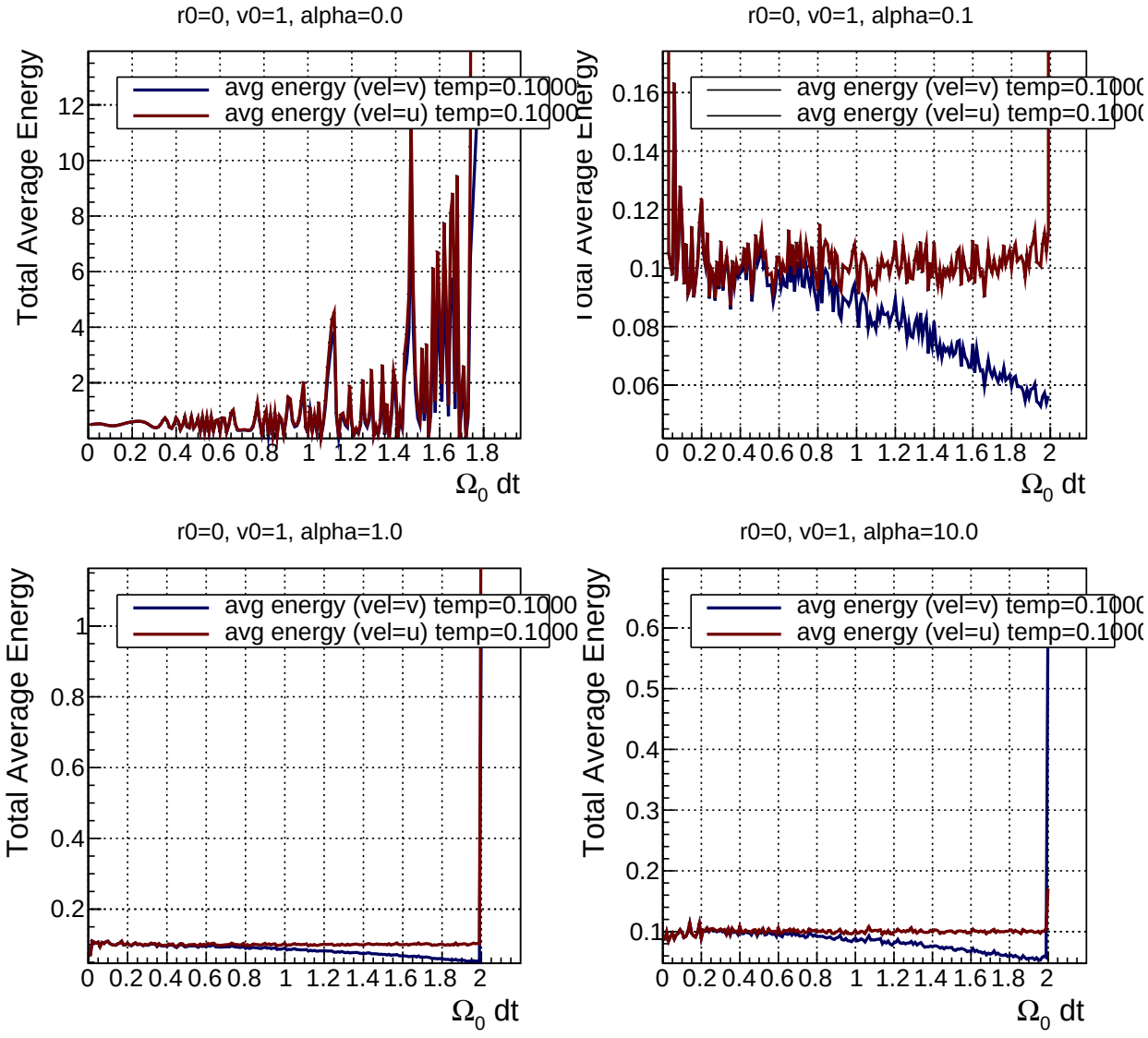

Figure 7: Average total energy (10^3 steps) at various friction coefficients (alpha), at initial conditions r_0=0 v_0=1

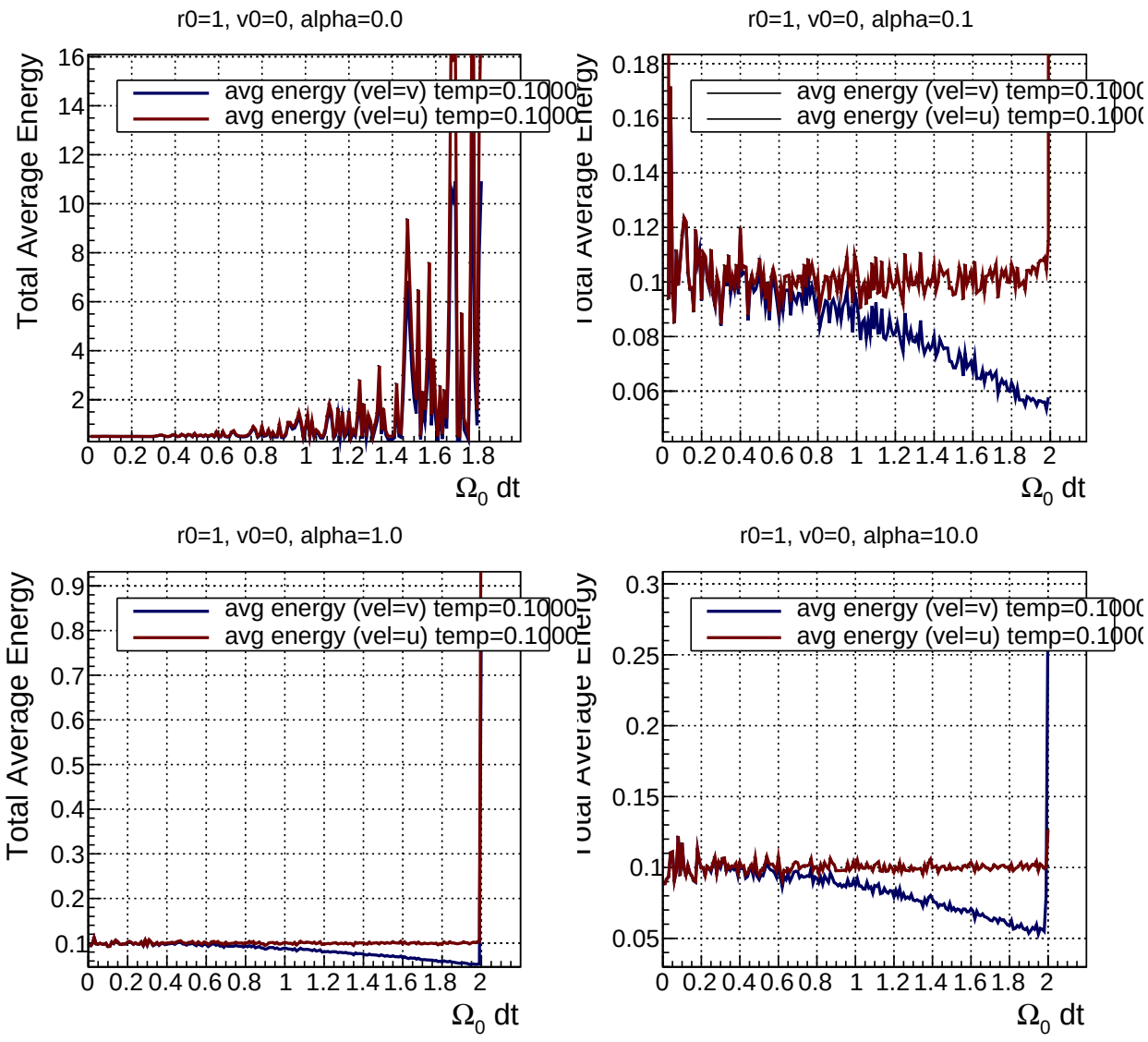

Figure 8: Average total energy (10^3 steps) at various friction coefficients and initial conditions r_0=1 v_0=0

Figures 7 and 8 offer a close up view of total average energies when $Omega_0 dt$ is less than 0.4. However,

3.3 Temperature=1

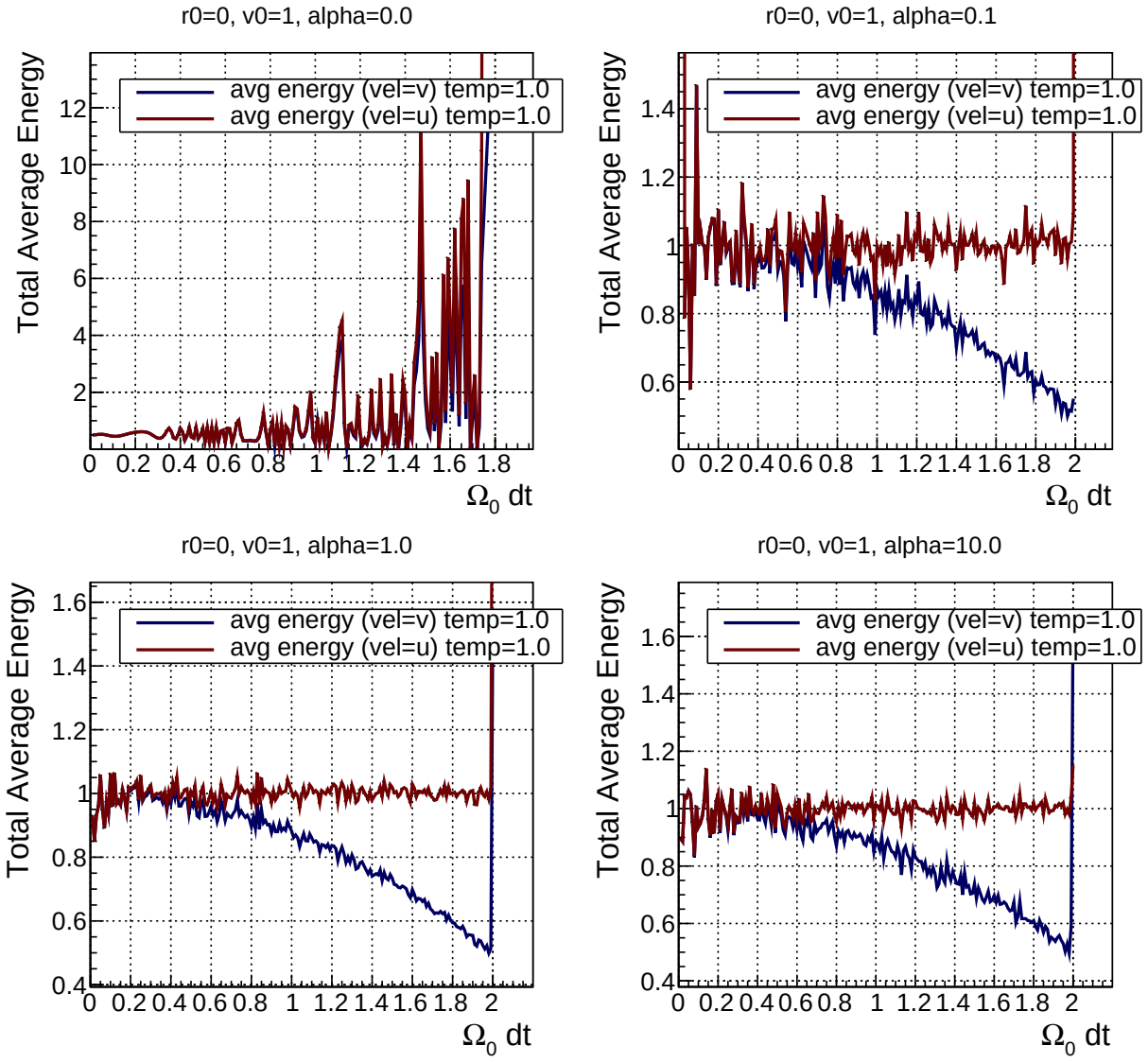

Figure 9: Average total energy (10^3 steps) at various friction coefficients and initial conditions r_0=0 v_0=1

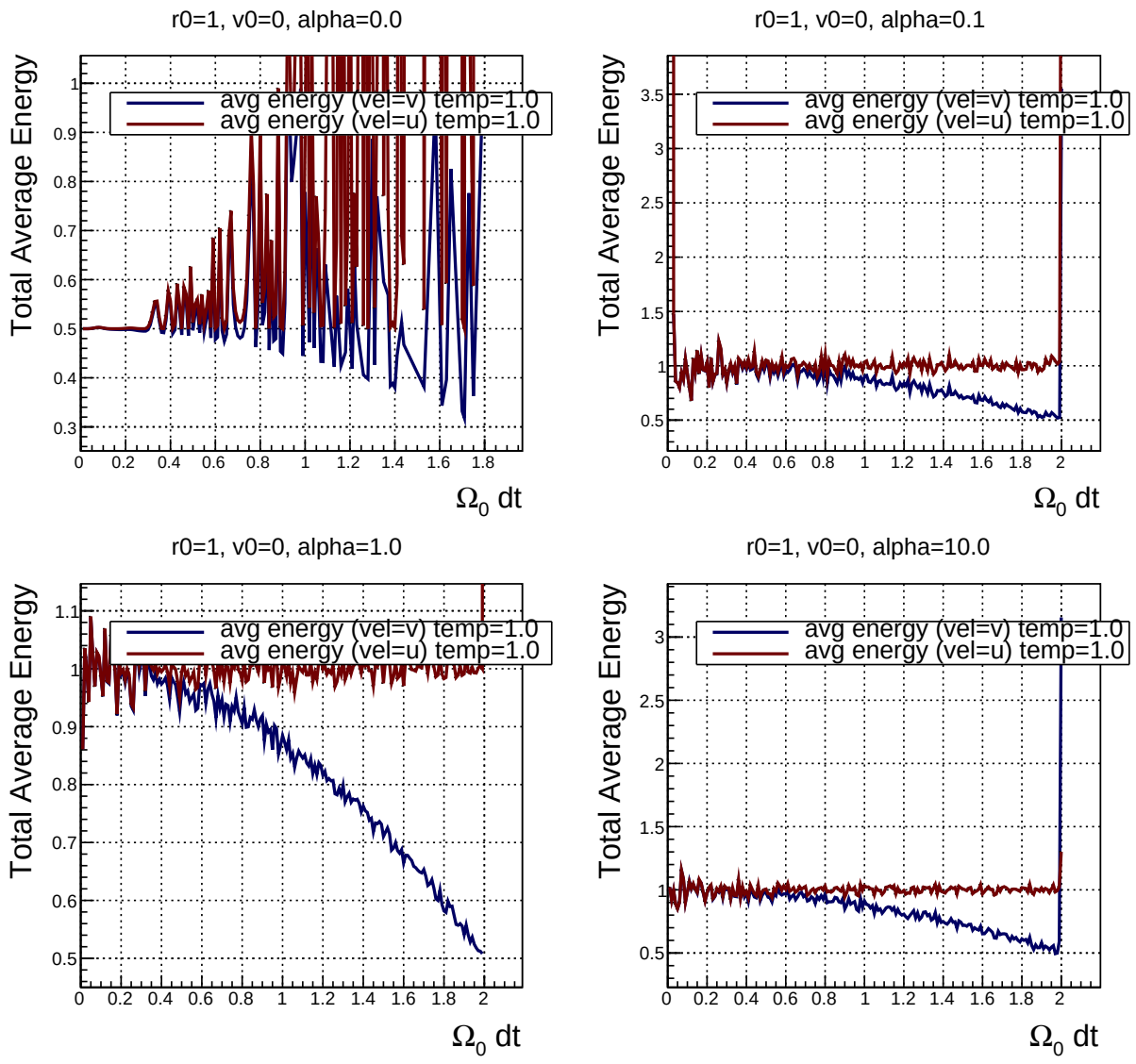

Figure 10: Average total energy (10^3 steps) at various friction coefficients and initial conditions r_0=1 v_0=0

3.4 Temperature=10

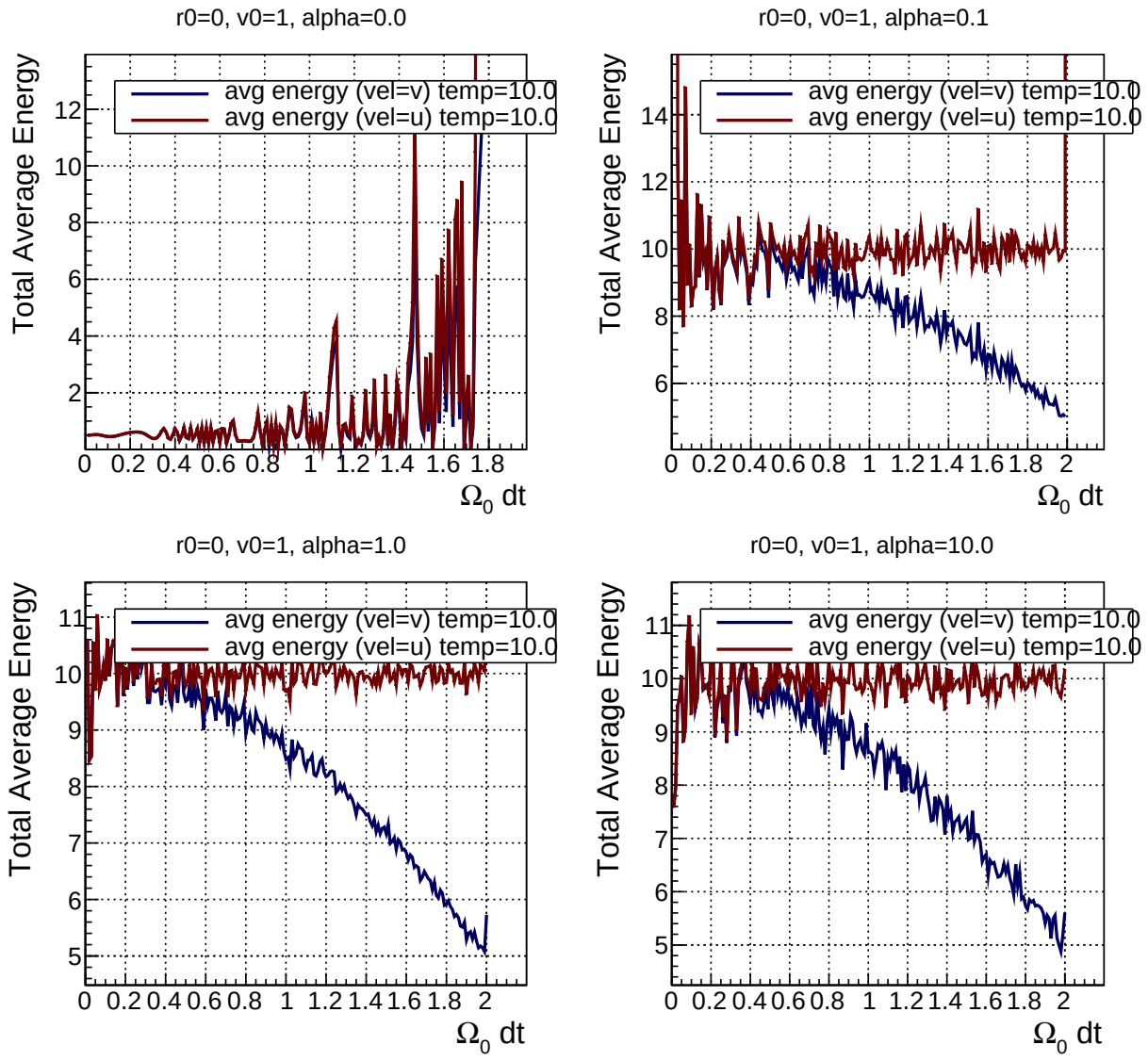

Figure 11: Normalized Temperature = 10

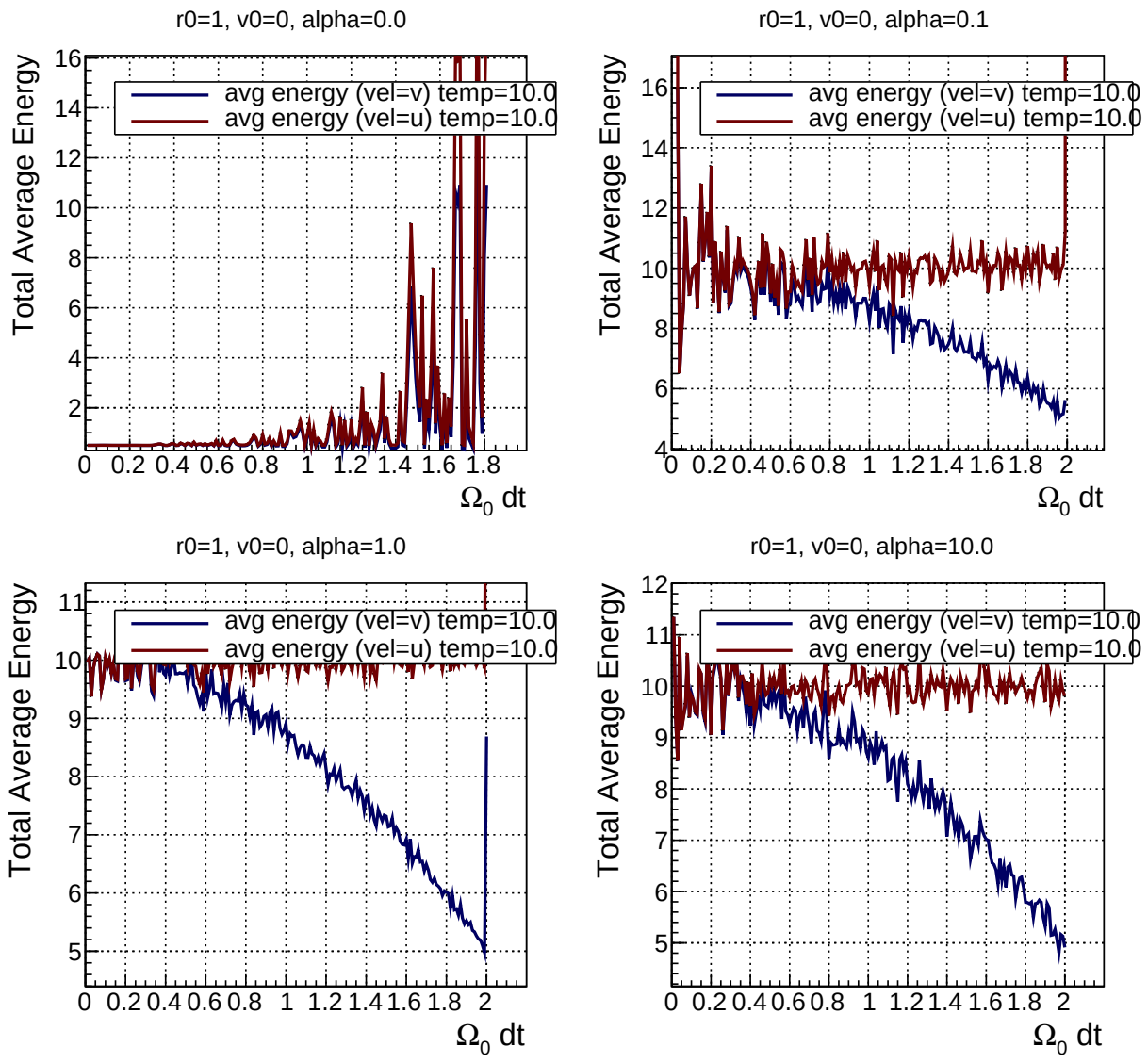

Figure 12: Normalized Temperature = 10

3.5 Discussion of Energy Results

Figures 7 to 8 illustrate energy proportional to kT, here we choose normalized value of $kT = 0.1$.
Figures 9 to 3.3 illustrate energy proportional to kT, here we choose normalized value of $kT = 1$.
Figures 11 to 12 illustrate energy proportional to kT, here we choose normalized value of $kT = 10$.

4 Task3: Opening Up the Boundaries

Alright now we have to open up the boundaries and deploy periodic boundary conditions. Meaning there needs to be a mirror image of the j-th particle with respect to our current i-th particle. So first we need to calculate the difference in their vectors

$$x_{ij} = x_i - x_j$$
$$y_{ij} = y_i - y_j$$

Impose the mirroring condition. Meaning for the i-th particle we check for the nearest neighboring particle with the following condition.

$$\text{if } x_{ij} > \frac{L_x}{2} \text{ then } x_{ij} -= L_x$$
$$\text{else if } x_{ij} \leq \frac{-L_x}{2} \text{ then } x_{ij} += L_x$$
$$\text{if } y_{ij} > \frac{L_y}{2} \text{ then } y_{ij} -= L_y$$
$$\text{else if } y_{ij} \leq \frac{-L_y}{2} \text{ then } y_{ij} += L_y$$

The code implementation of these conditions is show on the next page with a walk through explanation.

```
def main():
    for n_al in range(0,nal):
        for n_temp in range(0,ntemp):
            for n_dt in range(0,ndt):
                for n_time in range(0,nstep):
                    for i in range(0,natom):
                        #ith particle
                        for j in range(i,natom): #jth particle
                            #============================
                            #  Lennard-Jones Loop
                            #============================
                            xij=x[i]-x[j]
                            yij=y[i]-y[j]

                            #interbox collision
                            if i==j: #diagonal
                                lj_u_mat[i][j]      = 0 # u
                                lj_f_mat[0][i][j]   = 0 # fx
                                lj_f_mat[1][i][j]   = 0 # fy
                            else: #off diagonal
                                lj_u_mat[i][j]      = lj_potential(xij,yij)
                                lj_u_mat[j][i]      = lj_u_mat[i][j]
                                f_x,f_y             = lj_force(xij,yij)
                                lj_f_mat[0][i][j]   = f_x
                                lj_f_mat[1][i][j]   = f_y
                                lj_f_mat[0][j][i]   =-f_x
                                lj_f_mat[1][j][i]   =-f_y

                            # periodic boundary conditions
                            if xij>0.5*L:   xij-=L
                            if xij<=-0.5*L: xij+=L
                            if yij>0.5*L:   yij-=L
                            if yij<=-0.5*L: yij+=L
                            if i==j: #diagonal
                                lj_pb_u_mat[i][j]    = 0 # u
                                lj_pb_f_mat[0][i][j] = 0 # fx
                                lj_pb_f_mat[1][i][j] = 0 # fy
                            else: #off diagonal
                                lj_pb_u_mat[i][j]    = lj_potential(xij,yij)
                                lj_pb_u_mat[j][i]    = lj_pb_u_mat[i][j]
                                f_x,f_y              = lj_force(xij,yij)
                                lj_pb_f_mat[0][i][j] = f_x
                                lj_pb_f_mat[1][i][j] = f_y
                                lj_pb_f_mat[0][j][i] =-f_x
                                lj_pb_f_mat[1][j][i] =-f_y

                            #reenter box boundary
                            if x[j]>=L:     x[j]-=L
                            if x[j]<0:      x[j]+=L
                            if y[j]>=L:     y[j]-=L
                            if y[j]<0:      y[j]+=L
```

(1) Before computing the periodic boundary force, we first compute the interaction force between a i-j pair interaction. (2) then we check for the nearest jth particle image that lives outside the box (#periodic

boundary conditions). (3) Then we compute a new interaction force between i and the jth particle image. The matrices are used to hold each of these forces for a later calculation, (ie. the total force of the i-th particle). We need that force calculation to determine the evolution in the v^n.

What happens or what does the periodic boundary condition look like?
It can be seen that particles are confined within the box. Figures 13 and 14 show particles stay within the box limits ($0 < x < 20$ and $0 < y < 20$) verlet algorithm. So I need to fix the periodic boundary conditions.

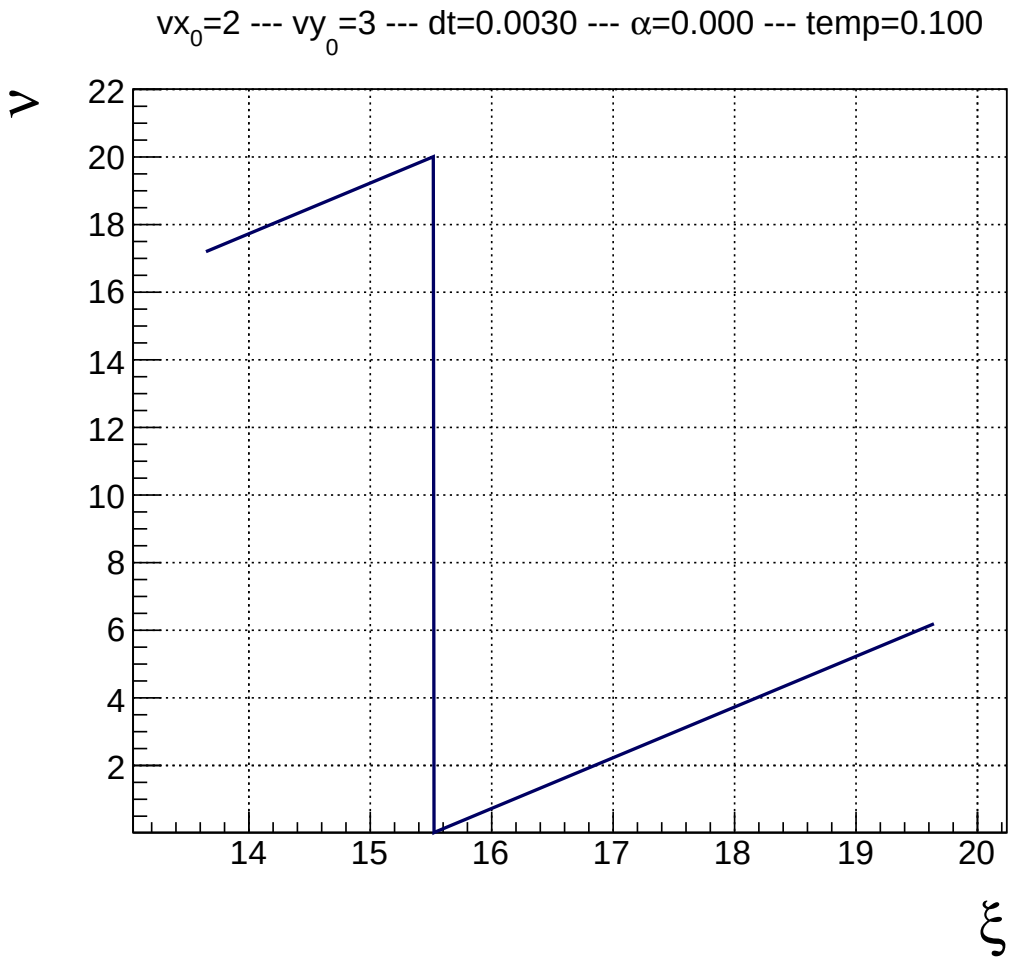

Figure 13: periodic bounds single particle

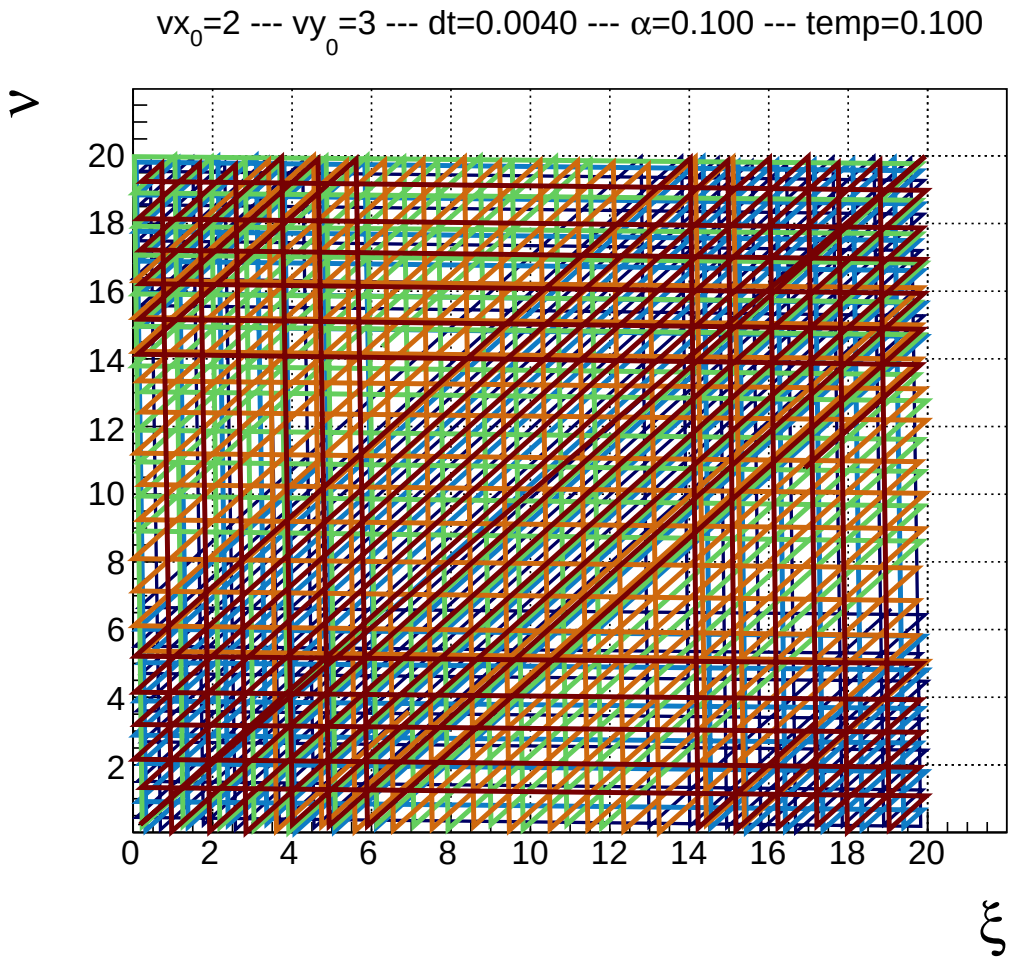

Figure 14: particle trace

4.1 Energy Analysis with Periodic Boundaries

4.1.1 Temperature=0.1

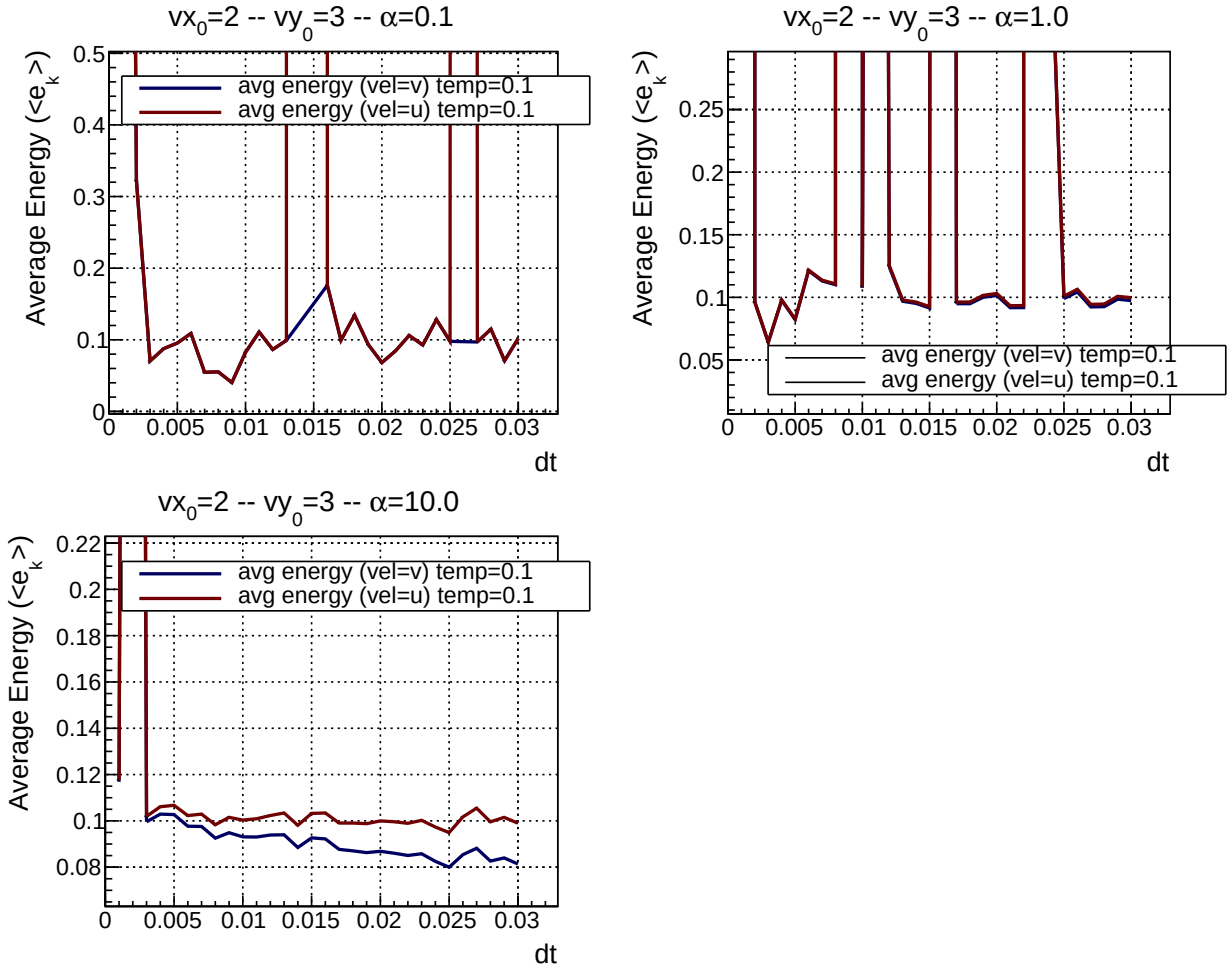

Figure 15: $0 < dt < 0.03$

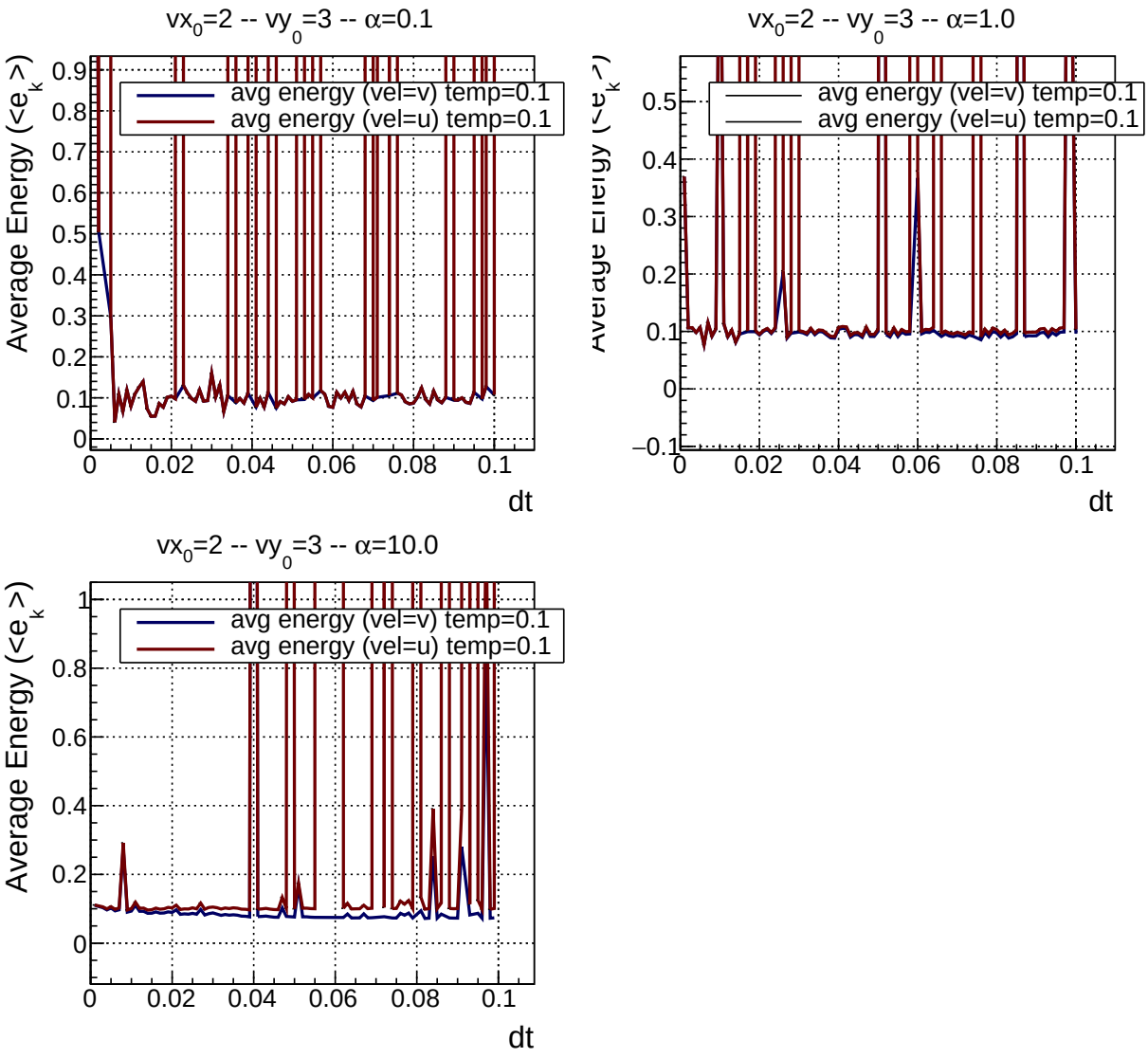

Figure 16: $0 < dt < 0.1$

4.1.2 Temperature=1

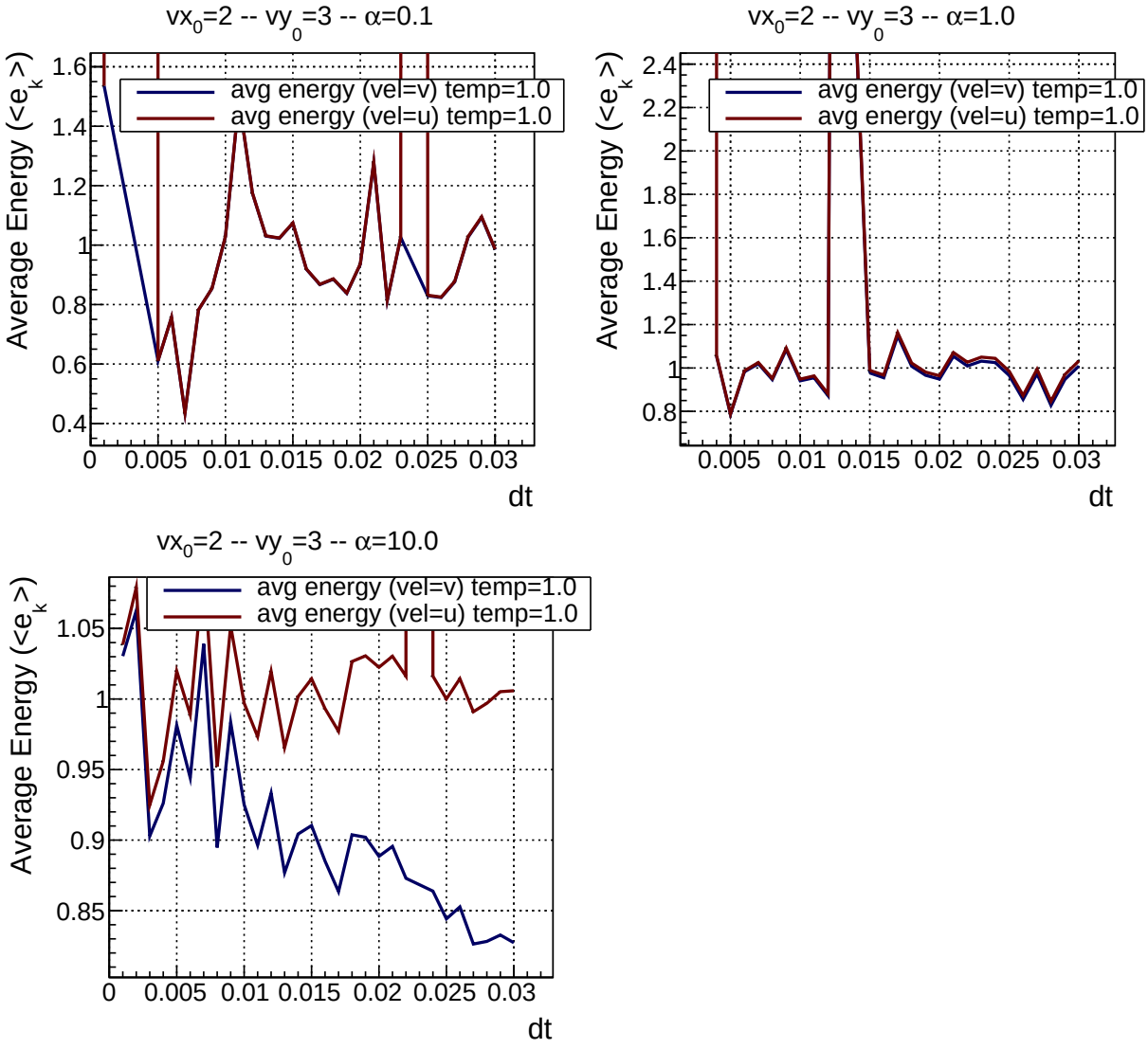

Figure 17: $0 < dt < 0.03$

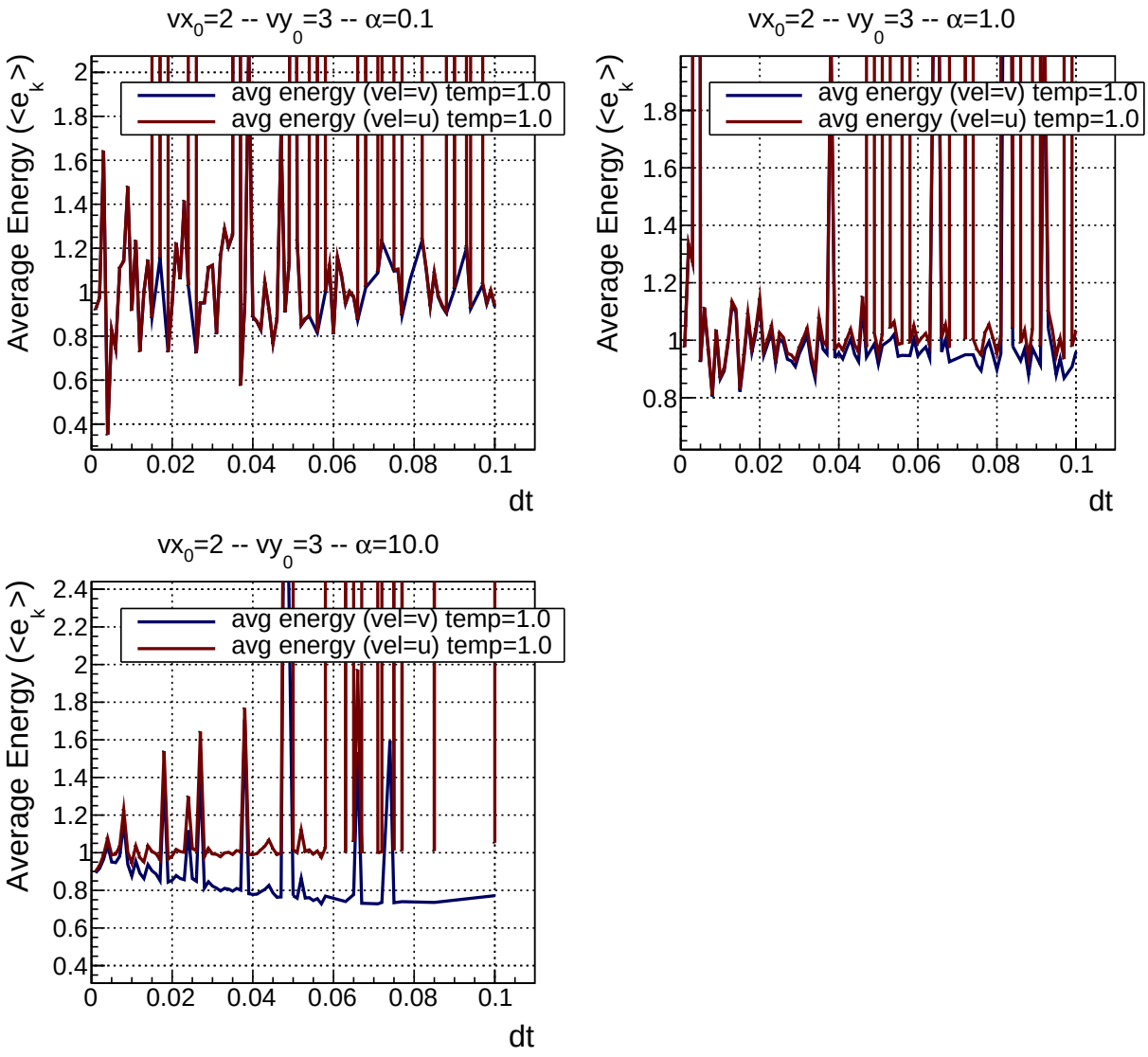

Figure 18: $0 < dt < 0.1$

4.1.3 Temperature=10

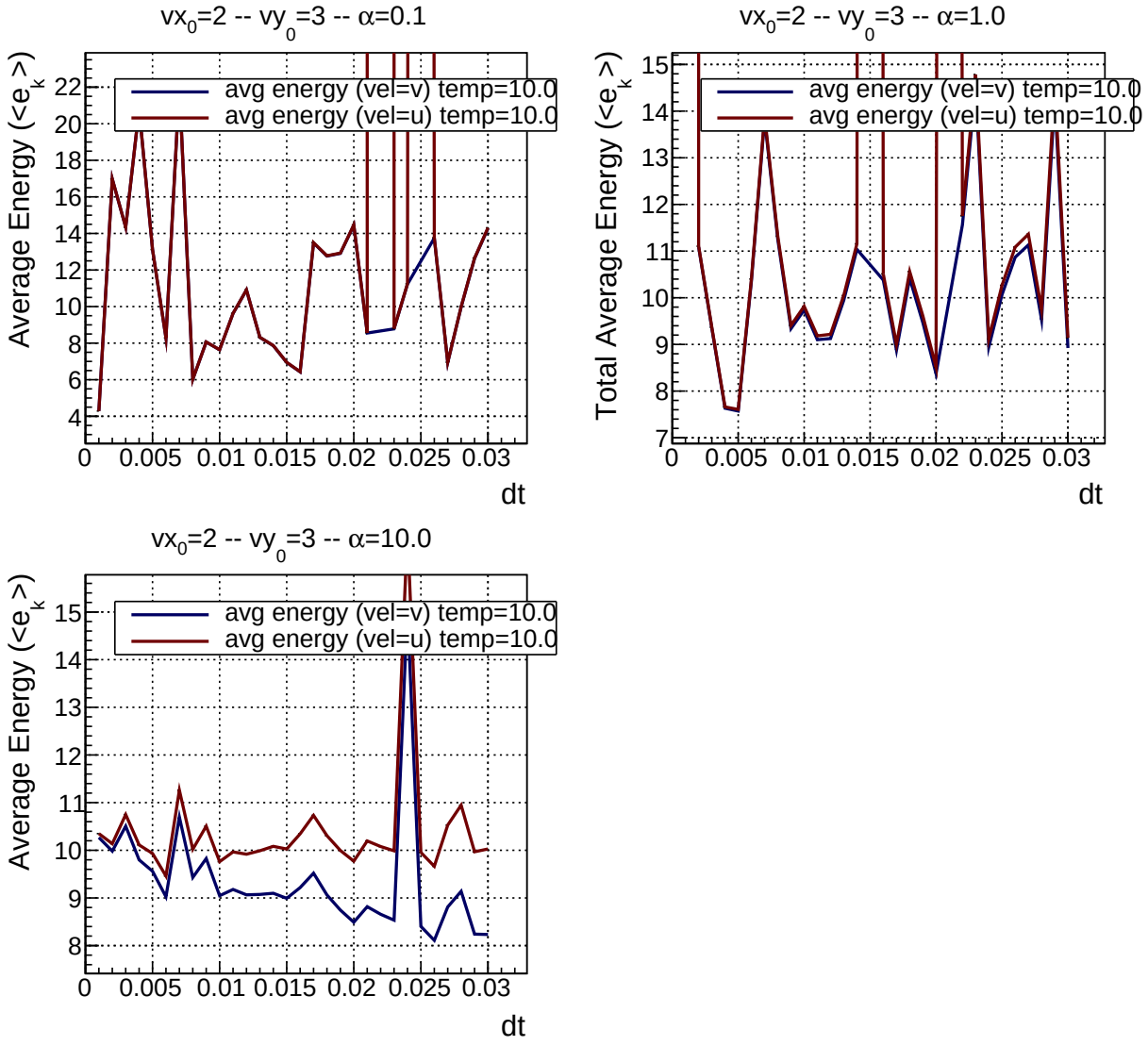

Figure 19: $0 < dt < 0.03$

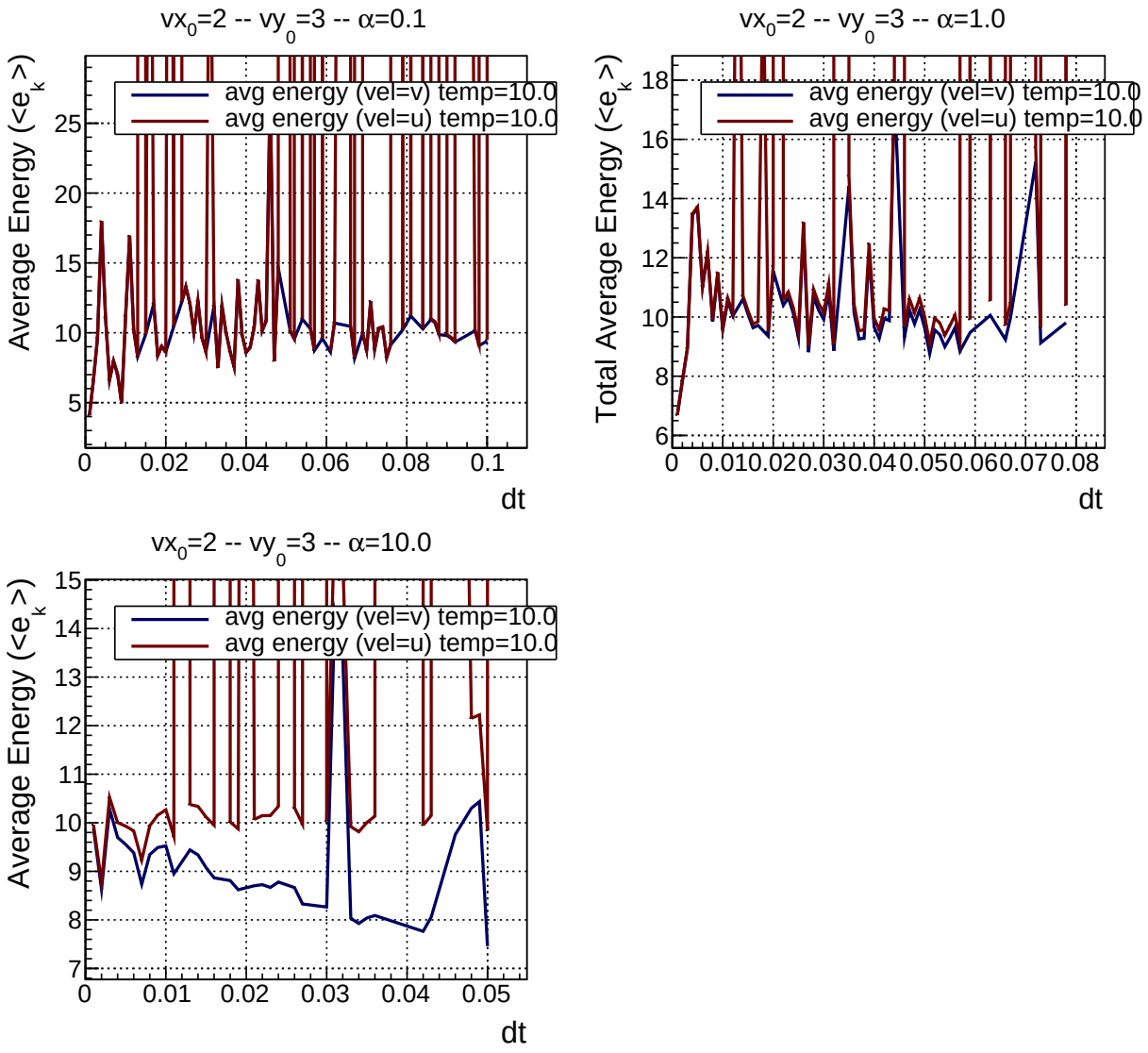

Figure 20: $0 < dt < 0.1$