Exercise 1:

Example1: n=8, timesteps=100, dt=0.1, mod=10

Example2: n=4, timesteps=1000, dt=0.01, mod=100

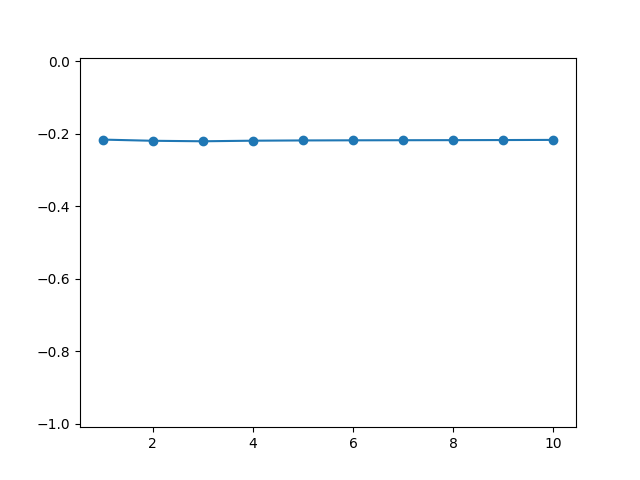
Exercise2:

The result of example 1 and 2:

See the attachment.

Exercise3:

The result of energy in example 1 is shown in the following figure.



Code:

double potential\_energy(int n, double3 \*\_\_restrict\_\_ x, double \*\_\_restrict\_\_ m)

{

  double e\_pot = 0.0;

  for (int i = 0; i < n; i++)

    for (int j = i + 1; j < n; j++)

    {

      double d0 = x[j][0] - x[i][0];

      double d1 = x[j][1] - x[i][1];

      double d2 = x[j][2] - x[i][2];

      double r2 = d0 \* d0 + d1 \* d1 + d2 \* d2;

      double r = sqrt(r2);

      e\_pot -= G \* m[i] \* m[j] / r;

    }

  return e\_pot;

}

double kinetic\_energy(int n, double \*\_\_restrict\_\_ m, double3 \*\_\_restrict\_\_ vb, double3 \*\_\_restrict\_\_ vf)

{

  double e\_kin = 0.0;

  for (int i = 0; i < n; i++)

  {

    double v0 = (vf[i][0] + vb[i][0]) / 2;

    double v1 = (vf[i][1] + vb[i][1]) / 2;

    double v2 = (vf[i][2] + vb[i][2]) / 2;

    double r2 = v0 \* v0 + v1 \* v1 + v2 \* v2;

    e\_kin += m[i] \* r2 / 2;

  }

  return e\_kin;

}

Exercise4:

Example: n=256, timesteps=1000, dt=0.01, mod=100

|  |  |
| --- | --- |
|  | GFlop/s |
| Vanilla AoS, B=32 | 10.9531 |
| AVX AoS, B=32, 1\*4 | 13.1133 |
| AVX AoS, B=32, 2\*4 | 11.0426 |
| intel SoA, B=32, width=4 | 12.3080 |
| intel SoA, B=32, width=8 | 11.4570 |